音楽におけるアノテーションとその応用に関する研究

梶　克彦

名古屋大学 工学研究科 計算理工学専攻

2004年2月

目 次

概要

音楽はただ聴くだけではなく、教育や娯楽など広い分野で利用されている。音楽に対して、様々な観点からの利用に有用な注釈や意味属性などのメタ情報 (アノテーション) を付与することができれば、高度な検索や要約などの情報サービスを容易に実現することができ、教育等における音楽利用を大いに促進することができるだろう。

本研究では、音楽に対してアノテーション情報を付与するシステムとして MiXA (MusicXML Annotator) を実装した。このシステムは、楽曲情報を XML で表現する MusicXML を使用し、Web ブラウザに楽譜を表示する。ユーザは楽譜に現れる音符や発想記号などの任意のオブジェクトに対してアノテーションを付与することができる。付与できるアノテーションの形式は、実現したいアプリケーションに応じて定義することが可能である。本システムを用いて多くのユーザからアノテーションを容易に収集することが可能となる。

また、本システムの評価実験を行い有効性を確認した。

本システムによって収集されたキーワードやコード進行などのアノテーションを用いて音楽コンテンツを検索するシステムと、曲の構成に関するアノテーションに基づいて曲を再構成するシステムを実装した。音楽検索システムは、キーワード検索に加え、コード進行による検索や曲の構成による絞り込み検索が可能である。音楽再構成システムは、曲の構成のアノテーションを利用し、ユーザが要求する構成に容易に変換することができるシステムである。

第1章　はじめに

1.1　デジタルコンテンツに対するアノテーション

近年、Web 上に氾濫する Web ページやマルチメディアコンテンツを、ただ見るだけでなく、高度に有効利用したいという欲求が高まってきた。そのため、現在それらのコンテンツに対してメタ的な情報を関連付けるアノテーションの研究が進められている。

その例としては、次世代の Web の形として提案された Semantic Web [12] の研究が挙げられる。元となる Web 上のコンテンツと、それに関連付けられた様々なメタ情報により、従来の Web では実現が困難と思われる高度な検索を可能にするものである。

また Web におけるアノテーションの研究には、他にセマンティック・トランスコーディング [13] がある。コンテンツとあらかじめそのコンテンツに対して関連付けられた文章の係り受け構造などのメタ情報を用いて、ユーザの特性や環境に合わせ、要約、翻訳、音声での読み上げなどを行う高度なコンテンツを動的に作り出すというものである。

このように、コンテンツを検索、変換する際に有利になるようなメタ情報を関連付けておけば、ユーザの要求に対して動的にコンテンツを作成することが可能となり、Web の利用をさらに高度にすることができる。

マルチメディアコンテンツに対するアノテーション研究の例としては、MPEG-7 [18] が挙げられる。ビデオコンテンツには、符号化の形式やシーンの内容、映し出されているオブジェクトが何であるかなど、非常に多くのメタ情報を潜在的に含んでいる。MPEG-7 により、ビデオコンテンツに潜在するこれらのメタ情報を記述する形式が定められている。ビデオコンテンツに十分なメタ情報が関連付けられていれば、例えば、誰々が何々をしているビデオはないか、といった検索や、コンピュータが自動で解析することが困難なオブジェクトやシーンの解析を容易に実現することができる。

1.2　デジタルコンテンツとしての音楽の記述形式

音楽もまた、インターネットで頻繁に流通しているデジタルコンテンツの一つである。デジタルコンテンツとしての音楽はテキストやビデオなどよりも多くの記述形式が存在する。

音楽の代表的な例として、MP3 [6] や MIDI [5] などが挙げられる。

MP3 は映像データ圧縮方式の MPEG-1 で利用される音声圧縮方式の一つであり、オーディオ CD 並の音質を保ったままデータ量を約 1/11 に圧縮することができる。しかし、MP3

は音声圧縮の形式であるため、楽曲としての情報、例えば、楽器の種類やメロディーなどは構造化されていない。

また MIDI はシンセサイザや音源とパソコンを接続して楽曲データをやりとりするための規格である。従来、楽曲の構造を解析し、変換するといった多くの研究には、MIDI が採用されてきた。しかし、MIDI は複雑な記述形式のため、構造解析や変換を容易に行うことはできない。また、MIDI は主に曲を演奏するための情報を記述するための形式であるため、音楽の代表的な表現形式である楽譜を生成するための十分な情報を含んではいない。

そこで、よりデジタルコンテンツとしての音楽を容易に扱える形式として、MusicXML [2] や WEDELMUSIC [1] といった、楽曲を XML [7] で記述する形式が提案された。XML は Java などの多くのプログラミング言語で容易に解析することができ、要素の挿入や削除などといった変換も容易に行うことができるというメリットがある。さらに、これらの形式は、MIDI などの従来使われてきた形式にも変換することができる。しかし、やはり MIDI と同様に、音楽を柔軟に扱うための情報がすべて含まれているわけではない。

1.3　音楽へのアノテーション

音楽はただ聴くだけではなく、娯楽、教育など非常に広い分野で利用されている。そして、その利用法も、BGM として利用したり、カラオケで歌うために伴奏だけで流したり、などのように多様である。このように音楽は、他のマルチメディアコンテンツに比べて非常に利用頻度、利用形態の多いのが特徴であるが、音楽は他に、絵画や彫刻などと同様に芸術作品としての一面を持つ。芸術作品は教育的に大きな意味を持っており、それらを高度に利用するための研究には高い意義があると考える。検索、分類、引用、要約などといった高度な利用を実現するためには、オリジナルのコンテンツが持つ情報だけでは不十分な場合が多いが、図 1.1 のように不足している情報をアノテーションとして補うことができれば、オリジナルのコンテンツとアノテーションにより高度な利用を実現することができる。

また、作品を制作する側としては、芸術作品を間違った意図で捕らえられることを避けるために、正しい情報を見る側、聞く側に伝えたいと考える場合があるだろう。そのような時にも、アノテーションとして、作品に自分の意図している正しい解釈の情報を付与することができれば、その作品を曲解されることを避けることができる。

また、複数のユーザがその作品に対して、感想などの情報をアノテーションとして関連付けることができれば、その楽曲に対するユーザの反応を知ることができ、よりよい音楽コンテンツを提供するモチベーションにつながるだろう。

しかし現在、音楽を含め芸術作品に対するアノテーションの研究はまだ十分に行われていない。

音楽に対してメタ情報を関連付ける研究としては、曲の雰囲気を自動で認識して検索に役立てる研究 [19] などが挙げられる。この研究は、曲の音やリズムの情報などを解析し、曲の印象というメタ情報を自動的に生成するというものである。しかしクラシックやポッ

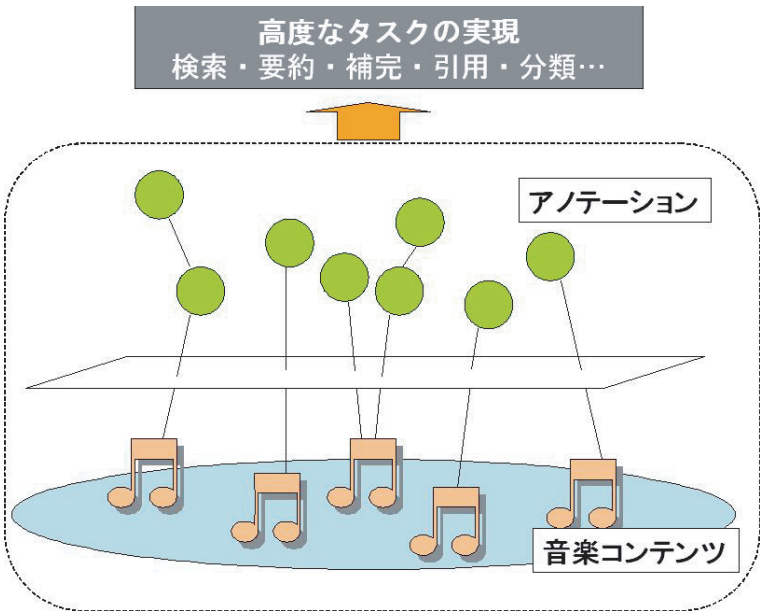

図 1.1: 音楽コンテンツとそのアノテーション

プスなど、曲にある程度の長さがある場合は、ひとつの曲の中に、例えば楽しい曲調の部分があったり、悲しげなメロディーの部分もあったりといったように、認識する箇所によって印象が異なる場合が多い。つまり、この研究のように曲全体のに対する印象のほかに、それぞれのパートごとに印象の情報が関連付けられていれば、より詳細な情報を用いた検索などが実現できるだろう。

このようにタイトルや作詞、作曲者名のような情報は、音楽コンテンツそのものに対する情報であるが、曲の印象の情報に限らず、曲を構成する部分要素に対して関連付けたいメタ情報が多く存在する。そのため音楽コンテンツに対するアノテーションは、曲全体に対するメタ情報を関連付けるだけでは不十分であり、曲を構成する要素、または要素の集合に対してもメタ情報を関連付けることができなくてはならない。

1.4　アノテーションシステムが備えるべき機能

現在のデジタルコンテンツに対するアノテーションシステムにはいくつかの問題点がある。

まず、実現したい利用法によって、どのようなアノテーションを付与する必要があるかが変わってくるため、単に雑多な情報を集めてしまうだけのシステムでは、収集されたアノテーションを二次利用する際に、有効に扱うことができない。つまり、アノテーションシステムを構築する際には、そのシステムで付与されるアノテーションを使ってどのよう

なことを実現したいのかが明確であり、そのためには元のコンテンツにどのような情報が足りないのかをあらかじめ明らかにしておく必要がある。

また、コンテンツにメタ情報を付与することを考える際には、拡張性を重視するべきである。もしアノテーションによって、全てのタスクに十分なあらゆる情報を記述することが可能であれば、あらかじめ元のコンテンツの記述形式を拡張し、初めからそれらの情報を含む形式にしてしまえばよく、アノテーションの必要性が乏しくなる。しかし現実には、テキスト、ビデオ、絵画など、どのコンテンツも、オリジナルのコンテンツに任意のタスクに十分な情報を記述することは不可能である。アノテーションに関しても同様であり、コンテンツに対する単独のアノテーションが、必ずしも二次利用に有効な形式となることができない場合がある。そのような場合を考慮し、アノテーションシステムは、コンテンツに付与されたアノテーションに対して、さらにアノテーションを記述することができることが望ましい。こうして階層化されたコンテンツとアノテーションを用いることで、より容易に様々なアプリケーションを実現することができる。

アノテーションをどのような人が付与するかについても、考慮が必要である。少人数の専門家がアノテーションの作業をするのであれば、スタンドアロンのシステムでも問題ないが、多くの人から幅広く集めたいメタ情報があるならば、ネットワーク性を重視しつつ、アノテーションを行うまでの手続きをできるだけ簡略化する必要がある。ネットワーク上で容易にアノテーションの作業を行う際には、Web ブラウザベースでシステムを構築すると、プログラムのインストール作業の必要も無く、ブラウザで特定のページに移動するだけでシステムが利用できる。さらに、Web ブラウザであれば、サーバの構築も容易であり、クライアントである Web ブラウザからスムーズにメタ情報を収集することができる。

また、アノテーションを行う対象であるコンテンツをどのような形式でユーザに提示し、何に対して情報を関連付けるのかを考慮する必要がある。ビデオコンテンツに対するアノテーションの場合、ユーザがビデオを見ている最中にアノテーションをする方法や、あらかじめシーン解析を行ったうえで、そのシーンに対してアノテーションしていく方法などが存在する。音楽に対するアノテーションシステムを考えた場合、曲を聴いている最中に情報を付与するを行うという時間軸に依存した手法や、楽譜に対してアノテーションを行う手法などがある。どのような情報を収集したいか、また対象とするユーザによって、提示するコンテンツの形式を決定する必要がある。

1.5　本論文の構成

本研究は、音楽に対するアノテーションシステムを構築すること、またそのアノテーションを用いた応用システムを実装することを目的とした。本論文では、2 章において、本研究で実装したシステム MiXA について、3 章と 4 章ではそれぞれ MiXA を用いて収集されたアノテーションを基にした検索と再構成のシステムについて述べる。5 章では MiXA の評価実験の結果と、それに対して行った考察について述べ、6 章と 7 章でそれぞれ本研究

のまとめと今後の課題について述べる。

第2章　音楽アノテーションシステム MiXA

音楽コンテンツの検索、再構成、要約といった高度利用を実現するためには、オリジナルのコンテンツの情報だけでは不十分な場合がある。そういった不足している情報をアノテーションとして付与していくことが必要となる。

そこで本研究では、音楽に対しアノテーションを付与するシステムとして MiXA (MusicXML Annotator)（ミキサと発音する）を実装した。以下で、本システムが備える機能とその実装を述べ、全体のシステム構成を説明する。

2.1　XML ベースのシステム

XML は複数のプログラミング言語でで容易に解析することができ、要素の挿入や削除などといった変換も容易に行うことができる。また、XML の検索言語である XPath [16] を利用することができ、コンテンツの特定の部分を指し示すことができる。そのため、アノテーションが関連付けられた音楽コンテンツの要素に容易にたどり着くことができる。このように、XML 形式には様々なメリットがある。

まず音楽コンテンツの記述形式として、MusicXML を採用した。MusicXML は図 2.1 のように、楽譜を形成するために十分な情報を記述できる。しかし、本研究で実現しようとしている検索や再構成システムを実装する際には、曲の印象や解説などの様々な情報が不足している。本システムは、MusicXML に対して不足しているアノテーションを付与するシステムである。

また、MusicXML に対するアノテーションの情報も XML 形式とした。アノテーションの内容を記述した XML には、どの音楽コンテンツの、どの要素集合に対して、どのような内容の情報が付与されているかが記述される。以後、MusicXML に対するアノテーションドキュメントをアノテーション XML と呼ぶ。

アノテーション XML の具体的な例は、付録 A-1 にある。以下にアノテーション XML の一部を抜き出して説明する。

```
<description
   id="description(kaji,/score-partwise/identification/creator[1]|
  /score-partwise/identification/creator[2])" x="428" y="83">
```

図 2.1: MusicXML の一部

```
  <source>
    <group>
      <object>/score-partwise/identification/creator[1]</object>
      <object>/score-partwise/identification/creator[2]</object>
    </group>
  </source>
  <information dataType="string">
    <string>このコンビで春が来た、おぼろ月夜、紅葉など有名な童謡を数多く発表して
います</string>
  </information>
 </description>
```

この XML の一部は、曲の解説に関するアノテーションの情報を表している。アノテー

ションが関連付けられている先の要素は、source タグ内に、XPath として記述されている。上記の XML の場合は、MusicXML に記述されている作詞者、作曲者に対するアノテーションであることがわかる。また、アノテーションの内容は information タグ内に記述されている。dataType 属性により解説の情報が、文字列型として記述されている。また、和音を表すコード型のアノテーションの場合、information タグ内は次のように記述される。

```
<information dataType="chord">
    <chord>
      <root>
        <step>A</step>
        <alter>1</alter>
      </root>
      <suffix>m7</suffix>
      <base>
        <step>D</step>
        <alter>1</alter>
      </base>
    </chord>
  </information>
```

コードには、root、suffix、base という複数の情報が同時に含まれている。そのため、アノテーションの記述形式もこのように構造化して表現する必要がある。

root は、そのコードを構成する元となる和音でありさらに step と alter という二つの情報から構成される。step は音高を表す、A、B、C、D、E、F、G のどれかの値をとる。また、alter はシャープやフラットといった情報を記述するためのもので、シャープの場合は 1、フラットの場合は-1 の値を取る。

suffix は、m(マイナー) や sus4(サスペンデッドフォース)、maj7(メジャーセブンス) といった接尾辞を記述するタグである。

base は、和音が展開形になった場合のベースとなる音を表す。root と同じく、step、alter から形成される。

上記の例では、A#m7/D#というコードの情報が表現されている。

本システムでは、このように文字列型やコード型といった複数のデータ型のアノテーションを作成することが可能である。データ型については、アノテーション定義 XML の節において詳しく述べる。

本システムではアノテーションの内容を、音楽コンテンツである MusicXML に直接書き込むということはしない。多くの人が一つのコンテンツに対するメタ情報を作成することを考えると、MusicXML に直接メタ情報を書き込んでしまうということは、その曲を作成していない人がコンテンツを改変することになってしまう。また、様々なメタ情報が付与

された MusicXML は、独自に拡張された仕様となってしまい、別のアプリケーションでその MusicXML を利用することができなくなってしまう。

以上の理由から、メタ情報を音楽コンテンツの内部に埋め込むということはせず、別の XML ドキュメントとして保存する。また、後述するマルチユーザによるアノテーションを可能にするために、各音楽コンテンツ対して、ユーザごとに個別に保存する。

2.2　アノテーション定義 XML

高度なサービスを実現するためのメタ情報は、それぞれのサービスに応じて、最適な内容や構造が異なる。そこで、実現したいサービスに応じて、柔軟に収集するアノテーションの形式を定義することができる仕組みを実装した。

XML ドキュメントに、どの要素に対して、どのような形式のアノテーションを関連付けを許すかといった定義を記述することで、システムは XML に記述されている情報を解析し、その記述どおりの形式のアノテーションを収集する。以降、この XML ドキュメントをアノテーション定義 XML と呼ぶ。

実際のアノテーション定義 XML の例は付録 A-2 に記述されている。この XML は、実現したいサービスとして、高度な検索、再構成のシステムを想定して記述したものである。以下にアノテーション定義 XML の一部を挙げ、説明する。

```
<annotation id="part" type="reference">

    <dataType>string</dataType>

    <expression color="#F0D044">構成</expression>

    <select>
      <item expression="イントロ" name="intro"/>
      <item expression="サビ" name="chorus"/>
      <item expression="間奏" name="bridge"/>
      <item expression="エンディング" name="ending"/>
      <item expression="Aメロ" name="verse-a"/>
      <item expression="Bメロ" name="verse-b"/>
      <item expression="Cメロ" name="verse-c"/>
    </select>

    <group>
      <item>
        <object minOccurs="0" maxOccurs="unbounded">note</object>
```

```
        <object minOccurs="0" maxOccurs="unbounded">rest</object>
        <object minOccurs="0" maxOccurs="unbounded">lyric</object>
      </item>
    </group>

  </annotation>
```

上記の XML は、曲の構成に関するアノテーションの定義を記述したものである。

まず、アノテーションタイプとして、リファレンスとリレーションのどちらであるかが annotation タグの type 属性として記述されている。リファレンスは、ある要素、または要素集合に対するプロパティを記述することができる。一方リレーションは、複数のある要素、または要素集合に関して、その要素間の関係を記述するものである。

dataType タグではアノテーションのデータ型として以下の 4 種類のどれかを指定する。

- string(文字列型)
- numeric(数字型)
- boolean(真偽型)
- chord(コード型)

データ型とは、作成されるアノテーションの内容である。コード型は、音楽における和音を記述するための型である。MusicXML には、現在それぞれの音符に対してコードを記述することができないが、コードは音楽を扱う際に頻繁に使用される情報であるため、本システムではコードをデータ型の一つとして実装した。

アノテーションとして作成することのできる内容を選択式にしたい場合には、select タグ内に選択することのできるアノテーションを記述することができる。select タグの存在は任意であり、無くてもかまわない。

本システムでは、楽譜上にそれぞれのユーザが作成したアノテーションの情報を重ね合わせて表示する。これらのオブジェクトのことを、以降アノテーションオブジェクトと呼ぶ。楽譜に表示する際のアノテーションオブジェクトの色、説明は expression タグに記述される。

どの要素に対してアノテーションを行うことができるかを記述したのが group タグである。

例えば、以下のように group が記述されていた場合、そのアノテーションは、音符、休符、歌詞を同時に複数含む要素集合に対して関連付けることができる。

```
<group>
```

```
    <item>
      <object minOccurs="0" maxOccurs="unbounded">note</object>
      <object minOccurs="0" maxOccurs="unbounded">rest</object>
      <object minOccurs="0" maxOccurs="unbounded">lyric</object>
    </item>
  </group>
```

アノテーションタイプがリレーションの場合は、group が複数記述することができ、その複数の group 間の関係のアノテーションを関連付けることができる。

アノテーション定義 XML の詳しい記述形式は、付録 A-3 にある、アノテーション定義 XML の書式を定義する XML Schema [8] に記述されている。アノテーションを収集しようとするサービス提供者は、このスキーマに従ってアノテーション定義 XML を記述する。

2.3　Web ブラウザを通したアノテーション

本システムではメタ情報を幅広いユーザから集め、その情報を様々に応用していくことを目的としている。しかしアノテーションシステムがスタンドアロンのシステムであった場合、ソフトウェアのインストール作業やサーバへの接続作業などに手間がかかるためユーザがそのシステムを利用する際の足かせとなり、結果多くの情報を集めることが困難である。

そこで、ネットワークにつながっている人なら煩雑な手続き無しでアノテーションが行えるように、Web ブラウザベースシステムの形態をとった。Web ブラウザを利用すると、サーバ側のシステムを容易に構築できるという利点もある。

多人数のユーザを対象とする場合、アノテーションの信頼性を考慮する必要がある。そのため前準備として、ユーザはあらかじめユーザ登録を行い、本システムを利用する際にはベーシック認証を通してログインする。誰がどのアノテーションを作成したかを全て保持するためである。付録 A-4 は、ユーザ登録により作成される個人プロファイルの XML の例である。

このようにアノテータを特定する機能を持っているという事実により、間違った情報の書き込みが減り、アノテーションの信頼度を高めることができる。

2.4　楽譜に対するアノテーション

楽曲の各要素 (タイトル、歌詞、音符、休符など) に対してアノテーションを行えるように、またユーザが、メタ情報を関連付ける部分を直感的に分かるようにするため、ユーザに提示する楽曲の形態は楽譜にした。楽曲の任意の部分にアノテーションができるようになれば、この部分はこのような意味がありますといったような解説や、この曲のこの部分

が気に入っているという感想、この部分はこのように弾きましょうといったような演奏支援の情報などをアノテーションとして関連付けることができる。これらの情報は、高度な検索や音楽教育などの実現を助けるだろう。

前述のように、ブラウザから本システムを利用できるようにするため、ブラウザに表示することができる形式の楽譜にする必要がある。そこで、本システムでは、楽譜の表示形式として、XML ベースの 2D ベクター画像記述言語である SVG (Scalable Vector Graphics) [14] を利用した。SVG により、図 2.2 のように楽譜をブラウザで表示することができ、さらに楽譜中に現れる音符などの要素に対するオペレーションに関しても、Web ページに動きや対話性を付加することができるスクリプト言語である JavaScript により記述することができるため、楽譜に対するアノテーションを実現することができる。

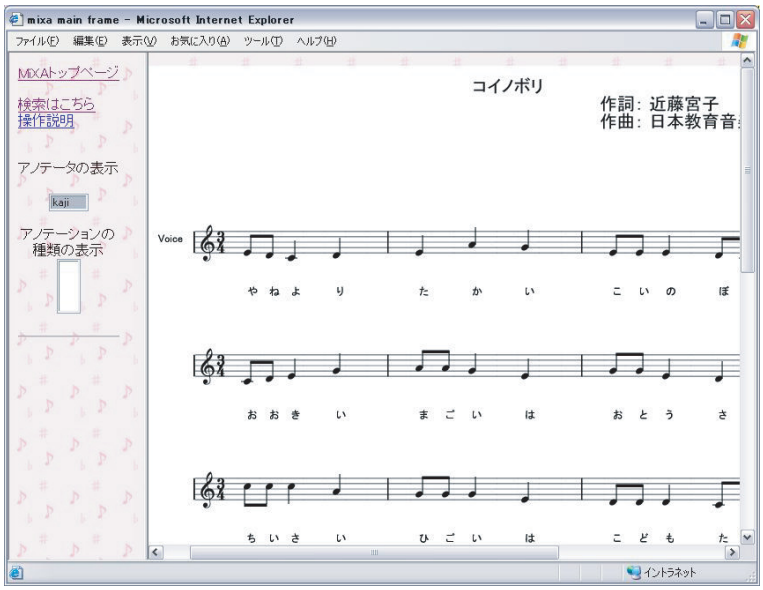

図 2.2: ブラウザに表示される楽譜

ユーザに提示される楽譜には、その音楽コンテンツに関して作成されたアノテーションのオブジェクトも重ねあわせて表示される。

それぞれのアノテーションオブジェクトには、アノテーションの内容に加えて、誰が作成したアノテーションであるか、楽譜に現れるどの要素に対するアノテーションであるかといった情報も記述されている。アノテーションオブジェクトにマウスカーソルを合わせると、図 2.3 のように、関連付けられている音楽オブジェクトも同時に強調表示し、一目でどの要素に対するアノテーションであるかを知ることができ、同時にそのアノテーションの詳細情報がポップアップで表示される。本システムは、ユーザ本人が作成したアノテーションオブジェクトと、他のアノテータが作成したアノテーションかにより、実行することのできるオペレーションに違いがある。表示される詳細のポップアップの色により、図

2.3、2.4 のように自分のアノテーションであるか、他人のアノテーションであるかを一目で識別することができる。

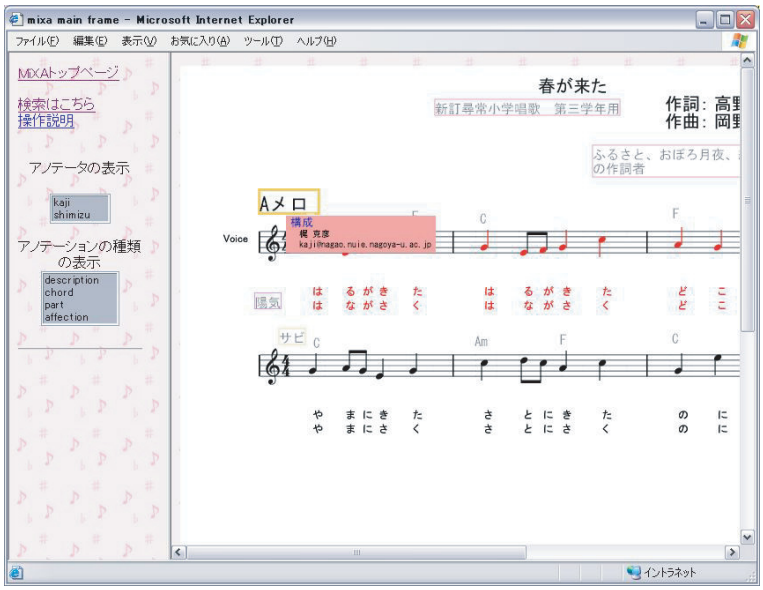

図 2.3: 自分のアノテーションオブジェクトの強調

具体的に、どのようにアノテーションを付与するかについて説明する。

アノテーションには大きく分けて、リファレンスとリレーションの二つのタイプがある。リファレンスは、ある要素、または要素の集合に対するプロパティを記述することができる。リレーションは、複数のある要素、または要素集合に関して、その要素間の関係を記述するものである。

リファレンスの場合のアノテーションは以下のように行う。ユーザはブラウザ上に表示された楽譜を見ながら、まず関連付けたい音符や休符、タイトル、発想記号といった要素をマウスの操作により選択する。

図 2.5 は実際に音楽オブジェクトを範囲選択した状態の図である。次に図 2.6 のようにどの種類のアノテーションを関連付けるかを決定し、その後具体的な情報を記述する。アノテーションの記述方式は、あらかじめアノテーション定義 XML に記述されているアノテーションのデータ型に依存する。

文字列型のアノテーションは、図 2.7 のように JavaScript のプロンプトから記入することができる。また、アノテーション定義 XML には、指定した内容から選択式のアノテーションを行うように記述することができるが、その場合、アノテーションは図 2.8 のようにメニューからの選択により容易に作成することができる。

一方、コード型のように複雑な構造を持つデータ型の場合は、図 2.9 のように、別ウィンドウからアノテーションの内容を記入する。

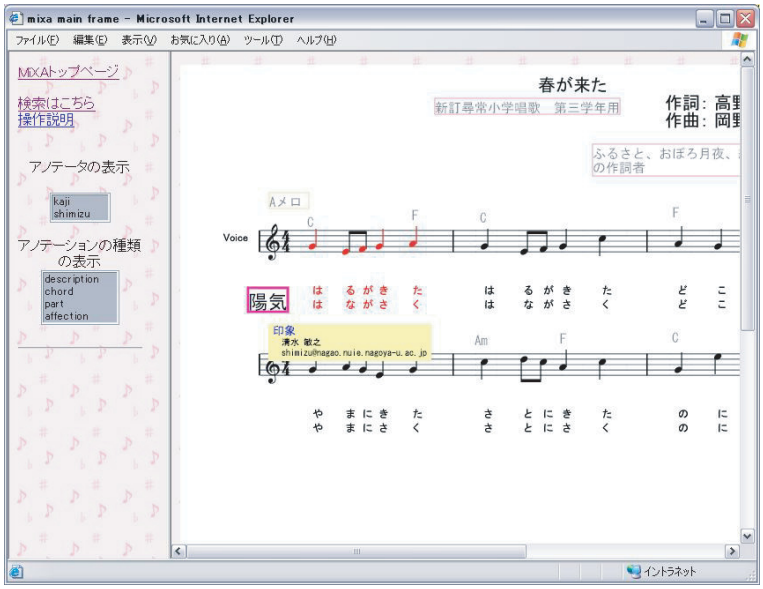

図 2.4: 他人のアノテーションオブジェクトを強調

次に、複数の要素間の関係を表すリレーションの場合のアノテーションについての手順を説明する。

リレーションタイプのアノテーションの説明の例として、二つの「解説」のアノテーションオブジェクトに対し、その二つのオブジェクト間の関係を記述する「解説の間の関係」というアノテーションを付与する場合を想定する。

「解説」と「解説の間の関係」のアノテーションは、具体的には次のようにアノテーション定義 XML において定義されている。

```
<annotation id="description" type="reference">

    <dataType>string</dataType>

    <expression color="#88F00F">解説</expression>

    <group>

      <item>
        <object minOccurs="0" maxOccurs="unbounded">note</object>
        <object minOccurs="0" maxOccurs="unbounded">rest</object>
        <object minOccurs="0" maxOccurs="unbounded">lyric</object>
      </item>
```

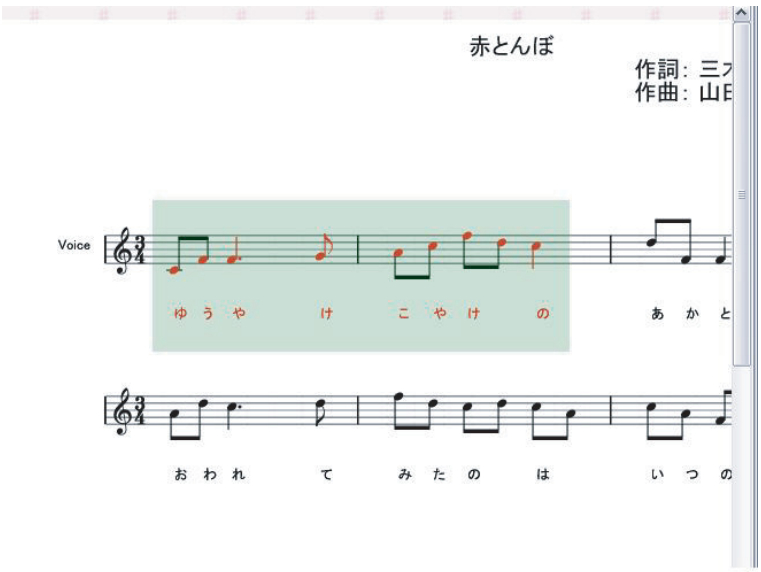

図 2.5: 音楽オブジェクトの範囲選択

```
    <item>
      <object>title</object>
    </item>

    <item>
      <object minOccurs="0">composer</object>
      <object minOccurs="0">poet</object>
    </item>

  </group>
</annotation>

<annotation id="relationOfDescription" type="relation">

  <dataType>string</dataType>

  <expression color="#272876">解説の関係</expression>

  <group>
```

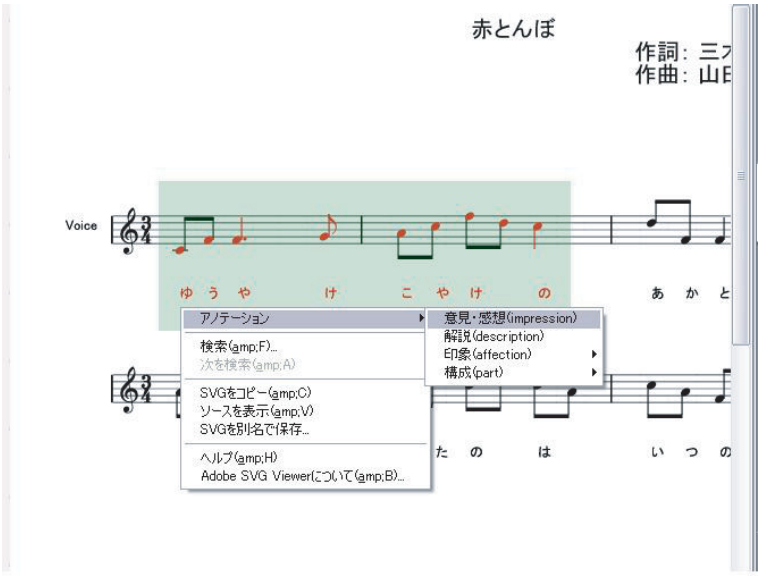

図 2.6: 範囲選択したオブジェクトに対するアノテーション

```
    <item>
      <object annotation="true">description</object>
    </item>
  </group>

  <group>
    <item>
      <object annotation="true">description</object>
    </item>
  </group>

</annotation>
```

「解説」のアノテーションは、音符、休符、歌詞の集合やタイトル、作詞作曲者に対して関連付けることのできる、文字列型のアノテーションとして定義されている。また、「解説の間の関係」のアノテーションは、annotation タグの属性として、アノテーションタイプを relation で宣言している。また、group タグが二つあり、それぞれの group タグに記述されている description は、「解説」のアノテーションの ID を示している。そのため、「解説の間の関係」は、「解説」のアノテーションを二つ選択し、その間の関係を記述することのできるアノテーションとして定義される。

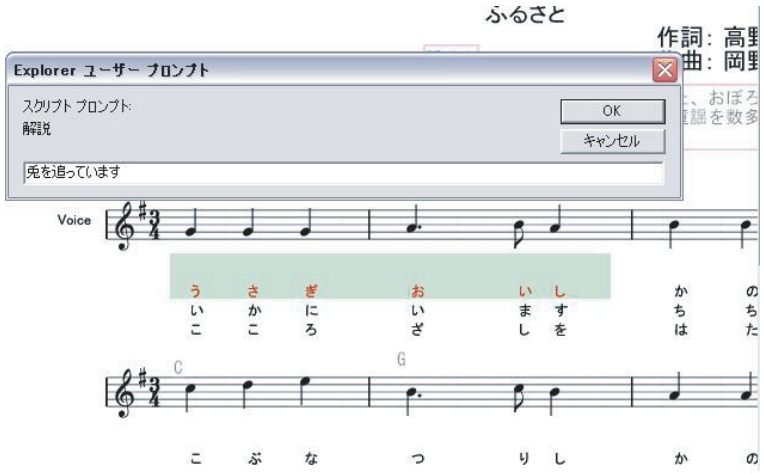

図 2.7: 文字列型のアノテーション

まず、図 2.10 のようにアノテーションを関連付ける「解説」のアノテーションオブジェクトを順に選択する。

選択された要素は、左側のフレームに表示されている。アノテーションは、楽譜に現れる音楽オブジェクト以外にも、このようにアノテーションオブジェクトに対しても関連付けることができる。アノテーションオブジェクトに対するアノテーションに関しては、アノテーションの階層化の節において詳細に述べる。

リレーションタイプののアノテーションを付与する要素を全て選択し終わったら、図 2.11 のように左フレームの「アノテーション」ボタンより、アノテーションを付与することができる。アノテーションを記述する方式は、リファレンスタイプのアノテーションの場合と同じく、アノテーションのデータ型に依存する。「解説の間の関係」のアノテーションは、文字列型として定義しているため、図 2.7 と同様に、アノテーションの内容をプロンプトから記述する。

リレーションタイプのアノテーションオブジェクトにマウスカーソルを合わせると、図 2.12 のように、どのオブジェクト間の関係を表しているかを強調表示する。

このように、作成されたアノテーションオブジェクトは楽譜上に表示される。ユーザはアノテーションオブジェクトを作成した後、そのオブジェクトをマウスでドラッグすることにより、楽譜上の見やすい位置へと移動させることができる。

作成したアノテーションを編集、削除したい場合は、該当するアノテーションオブジェクトにマウスカーソルを合わせ、右クリックメニューから実行することができる。編集、削除、移動といったオペレーションは、楽譜に現れる全てのアノテーションオブジェクトに対して行うことができる。

しかし、多くのユーザが同時にアノテーションを行う場合を考慮すると、アノテーショ

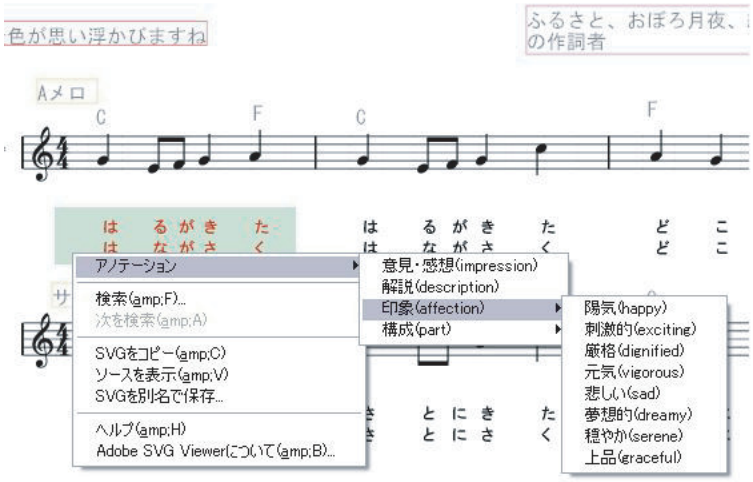

図 2.8: 選択式アノテーション

ンの保存を行う際に、削除、編集、移動といったオペレーションに関しては、自分が作成したアノテーションか、他者が作成したアノテーションかによって保存する内容を区別する必要がある。オペレーションの区別をしない場合、同じオブジェクトを同時に編集した時などに、アノテーションの情報に不整合がおきてしまうからである。そこで、アノテーションを保存する際のポリシーを以下のように決定した。

- 自分が作成したアノテーションオブジェクトは楽譜上の移動や編集、削除した結果をサーバに保存することができる。
- 他人のメタ情報は移動や削除はできるがその変更結果は保存できない。

作成したメタ情報をサーバに保存する場合には、マウスをどのオブジェクトにも合わない状態で、右クリックメニューからアノテーションの保存を選択する。

本システムでは一つの音楽コンテンツに対して複数の人がアノテーションを行い、ユーザが他人の作成したメタ情報を見られるようにするため、他のアノテータが作成したアノテーションオブジェクトも全て楽譜上に表示されている。しかし、楽譜上には様々な情報が混在する状態となり、アノテーションをスムーズに行えなくなってしまう場合がある。

そのような場合を考慮し、本システムはアノテーションのフィルタリングを行うことができるようにした。左側のフレームから、表示したいアノテータの ID、アノテーションの種類を選択すると、該当するアノテーションオブジェクト以外のオブジェクトが画面上から無くなり、楽譜や注目すべきアノテーションオブジェクトを見やすくすることができる。

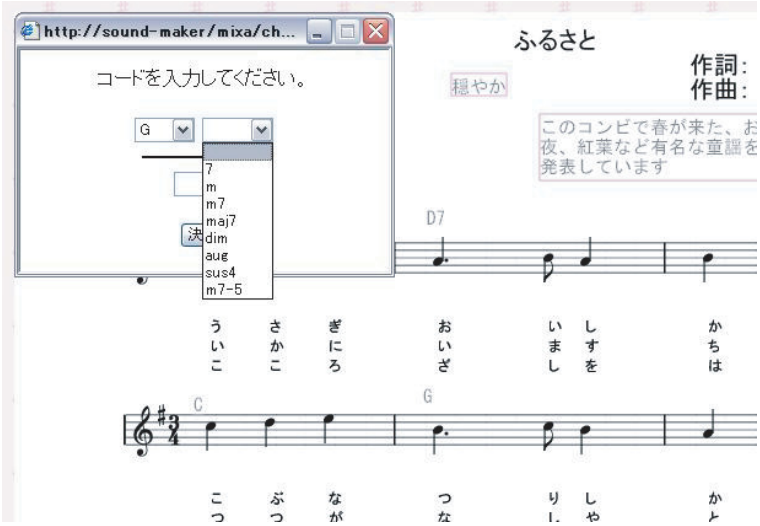

図 2.9: コード型のアノテーション

2.5　アノテーションの階層化

アノテーションの情報に拡張性を持たせるために、アノテーションオブジェクトに対してアノテーションを関連付けることができるようにした。このことで、コード進行のように、一つの単純なアノテーションでは記述しきれないような情報でも、階層化したアノテーションの集合により記述することが可能である。

アノテーションオブジェクトに対するアノテーションは、図 2.13 のように、対象となるアノテーションオブジェクトにマウスカーソルを合わせ、右クリックメニューから、関連付けたいアノテーションを選択する。

アノテーションオブジェクトに対するアノテーションにより、掲示板のようなユーザ同士によるインタラクションが可能になる。そのため、本システムによりユーザ同士が音楽コンテンツに対して意見を言い合う場としても利用することができる。

アノテーションが多いほど、検索などの応用システムの精度を高めることができるため、本システムは、多くのユーザからアノテーションを広く収集することを目的としている。このように本システムによりユーザ同士の意見交換を行うことができると、より多くのユーザが本システムを利用することになるため、多くのアノテーションを収集することができるようになる。

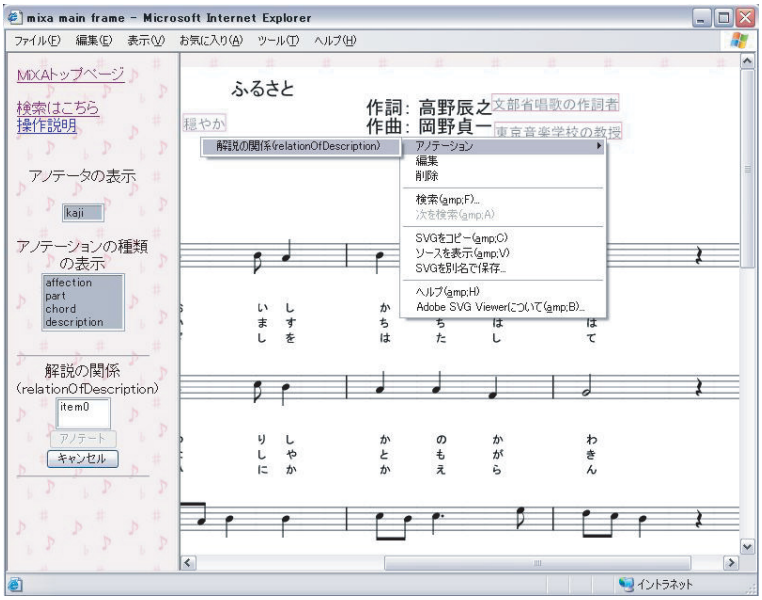

図 2.10: リレーションタイプのアノテーション 1

2.6　システム構成

図 2.14 は、本システムのシステム構成の概要を示している。この図のように、ユーザは Web ブラウザを通して、複数のデータベースを管理する Web サーバとやり取りを行うことによってアノテーションを作成し、共有することができる。詳細は以下の通りである。

あらかじめ Web サーバ側に XML データベース (Xindice [15]) を用意し、その XML データベースに音楽コンテンツである MusicXML を保存しておく。また、本システムを利用するユーザのプロファイルを記述した個人プロファイル XML を、同じく XML データベースに保存しておく。音楽コンテンツに対するアノテーションの XML ドキュメントも音楽コンテンツや個人プロファイルと同様に XML データベースに保存されることになる。Web サーバには、Java Servlet [17] を使用した。

ユーザは Internet Explorer からユーザ名とパスワードを求めるベーシック認証を通して図 2.15 の本システムのトップページにアクセスする。

登録されている曲の一覧から、または後述する検索システムの検索結果一覧から、アノテーションを作成したい曲を選択すると、Web サーバは要求された曲の MusicXML を XML データベースから取り出す。さらにその MusicXML をブラウザで表示することのできる SVG 形式の楽譜に変換する。作成された楽譜の各音楽オブジェクトには、元の MusicXML のどの部分を指すオブジェクトであるのかを指し示す XPath が内部的に記述されている。

MusicXML から楽譜を作成する際、該当する曲に関するアノテーションを XML データベースから検索し、アノテーションが検索された場合はそれらの情報をアノテーションオ

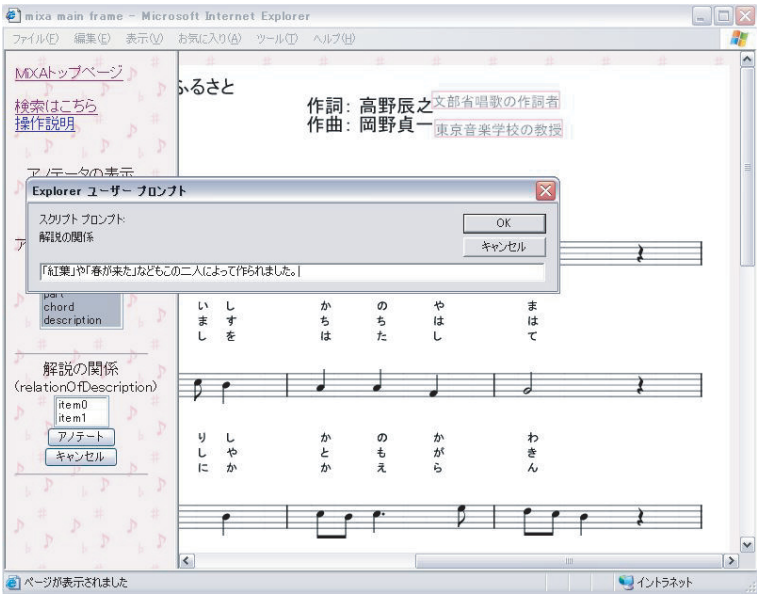

図 2.11: リレーションタイプのアノテーション 2

ブジェクトとして楽譜に視覚的に重ね合わせてブラウザに表示する。また、それぞれのアノテーションオブジェクトを作成したアノテータの個人情報を XML データベースから取得しておく。

ユーザはブラウザに表示されている楽譜上でマウスとキーボードの操作によりアノテーションを行う。

ユーザがアノテーションの保存を選択すると、そのユーザが作成したアノテーションの内容や、楽譜上の座標などをサーバに送信する。サーバは、1 ユーザ、1 コンテンツにつき一つの XML ドキュメントを作成し、ユーザがその曲に対して作成したすべてのアノテーション情報を記述し、XML データベースに保存する。

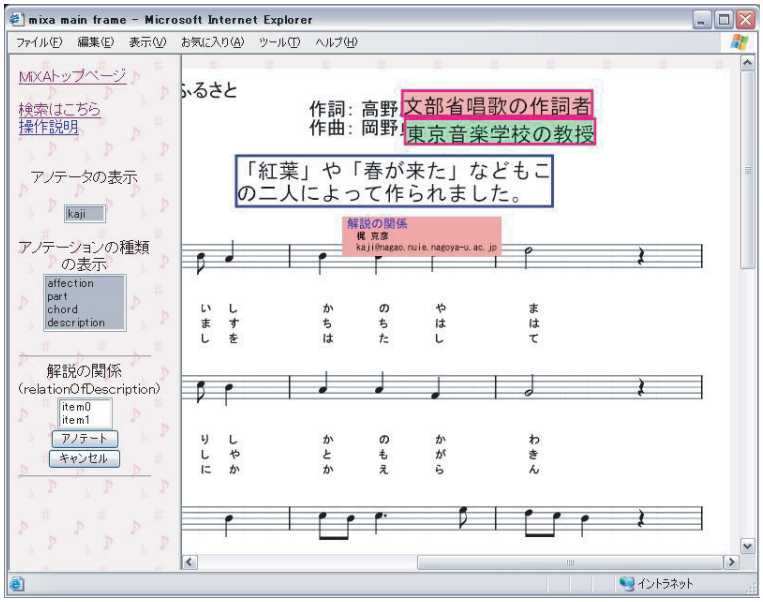

図 2.12: リレーションタイプのアノテーションオブジェクトの強調

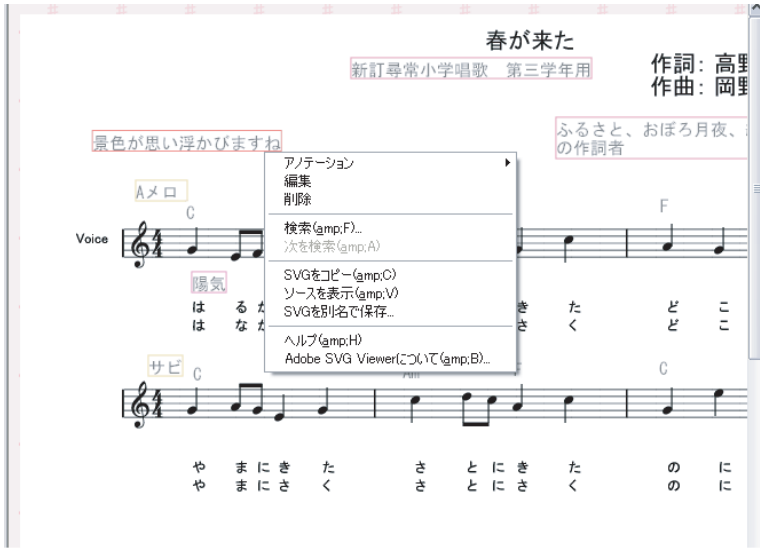

図 2.13: アノテーションオブジェクトに対するアノテーション

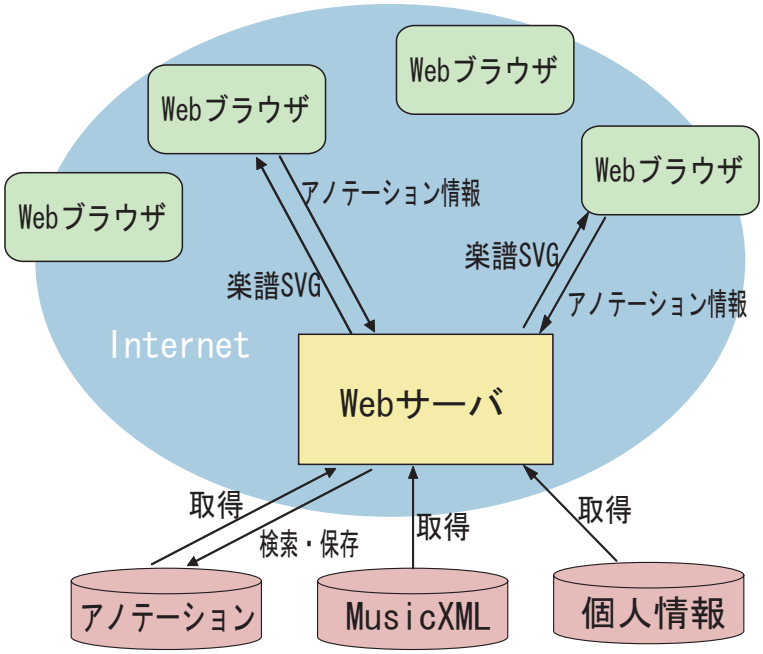

図 2.14: MiXA のシステム構成

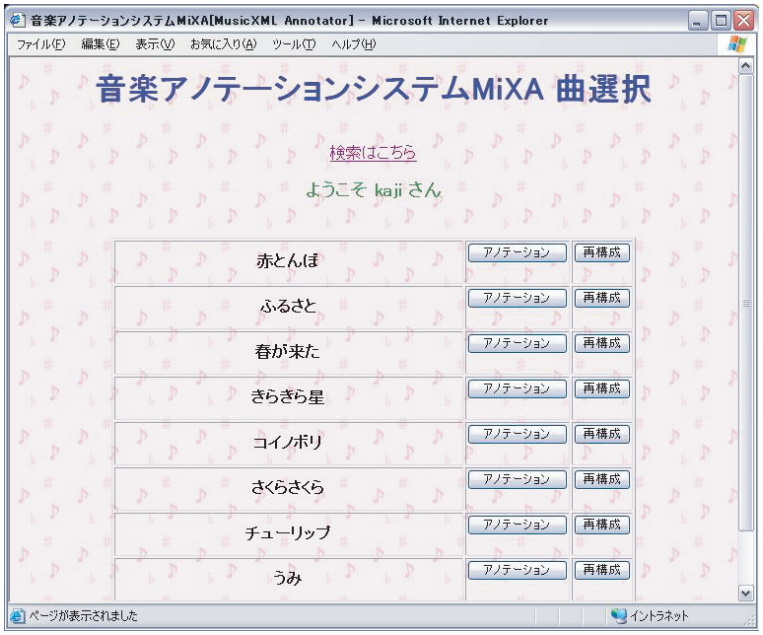

図 2.15: MiXA 曲選択画面

第3章　アノテーションの応用システム

MiXA によるアノテーションは、一般には実現が困難と思われる様々な情報サービスの実現を容易にすることができる。各音楽コンテンツに対する多くの詳細なメタ情報を集めることができるので、検索に関しては特に大きく精度の改善を期待することができるだろう。また曲を再構成したり、曲のある部分を別の曲の途中に付け加えたりといった、コンピュータの自動解析だけでは実現が難しい、曲の構造を利用した高度なアプリケーションに関しても、コンテンツに詳細なメタ情報が関連付けられていれば、精度の高い解析結果を利用することができる。

本研究では、アノテーションを用いた応用システムの例として、音楽検索、音楽再構成のシステムを実装した。以下に詳細を述べる。

3.1　音楽検索システム

3.1.1　音楽検索のためのアノテーションの収集

高度な検索システムを実現するための前準備として、MiXA により意見・感想、解説、印象、コード、曲の構成という四種類のアノテーションを収集することにした。アノテーション XML には、表 3.1 のようにそれぞれのアノテーションを定義した。

表 3.1: 検索システムのためのアノテーション定義

名前	データ型	関連付けることのできるオブジェクト
意見・感想	文字列型	音符・休符・歌詞の集合、 タイトル、作詞・作曲者の集合、意見・感想オブジェクト
解説	文字列型	音符・休符・歌詞の集合、タイトル、作詞・作曲者の集合
印象	文字列型	音符・休符・歌詞の集合、タイトル
曲の構成	文字列型	音符・休符・歌詞の集合
コード	コード型	音符、休符、歌詞の集合

「意見・感想」は、多くのユーザから、音楽コンテンツの各部分に対して自由に文字列型のコメントを書き込んでもらうためのアノテーションである。ある人の意見に対して、

別の人が意見を述べる場合を考慮して、「意見・感想」のアノテーションオブジェクトに対しても「意見・感想」のアノテーションを関連付けることができるようにした。

「解説」は、音楽コンテンツのタイトル、作詞作曲者に対してその曲の説明をしたり、音符や歌詞などに対して、その部分にはどのような意味あるかなどを文字列型として記述することができる。

「印象」は、曲の各部分に対し、ユーザがどのような印象を受けたかを文字列型として記述することができる。印象のアノテーションは、陽気、刺激的、厳格、元気、悲しい、夢想的、穏やか、上品の 8 種類から選択することができる。印象のアノテーションの選択候補とした 8 種類の印象語は、Hevner の印象語群の定義 [10] によった。Hevener は、これら八つの印象語によって、音楽の印象を分類することができると定義した。

「曲の構成」は、ポップスなどに見られるイントロやサビといった、その曲が大まかに見てどのように構成されているかを記述するためのアノテーションである。アノテーションとして付与することのできる情報は文字列型で、イントロ、間奏、エンディング、サビ、A メロ、B メロ、C メロから選択する。

「コード」は、曲の各部分が、どのような和音構成であるかを記述する。コードのデータ型は、あらかじめ本システムで実装したコード型を使用する。コード型には、root、suffix、base という情報が含まれており、例えば Am/E というコードの場合、root が A、suffix が m(マイナー)、base が E となる。コードのアノテーションは、音符、休符、歌詞の集合に対して関連付けることができる。

上記の定義でアノテーション定義 XML を記述した。付録 A-2 は、実際に検索システム実現のために MiXA に適用したアノテーション定義 XML である。

以上の五つのアノテーションを、MiXA により音楽コンテンツの各要素に付与する。本研究では、これらのアノテーションを長尾研究室の学生が作成した。ただし、コードなどの専門知識が必要となるアノテーションについては、信頼できる情報を作成するために、知識のあるユーザが作成した。今回はアノテーションの正確さについては厳密に考慮せず、アノテーションの内容はすべて正しい情報であるとみなすことにする。

こうして MiXA を利用して音楽コンテンツに対するアノテーションを収集することができた。以下に、これらの情報を用いて音楽検索を行うシステムの詳細を述べる。

本検索システムでは、音楽コンテンツである MusicXML に記述されている、タイトルや作詞・作曲者の情報に加えて、アノテーションとして音楽コンテンツに関連付けられているメタ情報を用いて、キーワード検索を行うことができる。また、コード進行による検索や、曲の構成による絞り込み検索をすることができる。本検索システムは Web ブラウザベースで実装されているため、ユーザは Web ブラウザを通して検索を行うことができる。

3.1.2　キーワード検索

キーワード検索の流れを説明する。

まずユーザがタイトル、作詞者、作曲者、意見・感想、解説の情報を図 3.1 の Web ブラウ

ザのフォームから検索の要求を行う。Web サーバはその検索要求を受け取り、以下のようにして該当する音楽コンテンツを検索する。Web サーバには、MiXA と同様 Java Servlet を利用した。

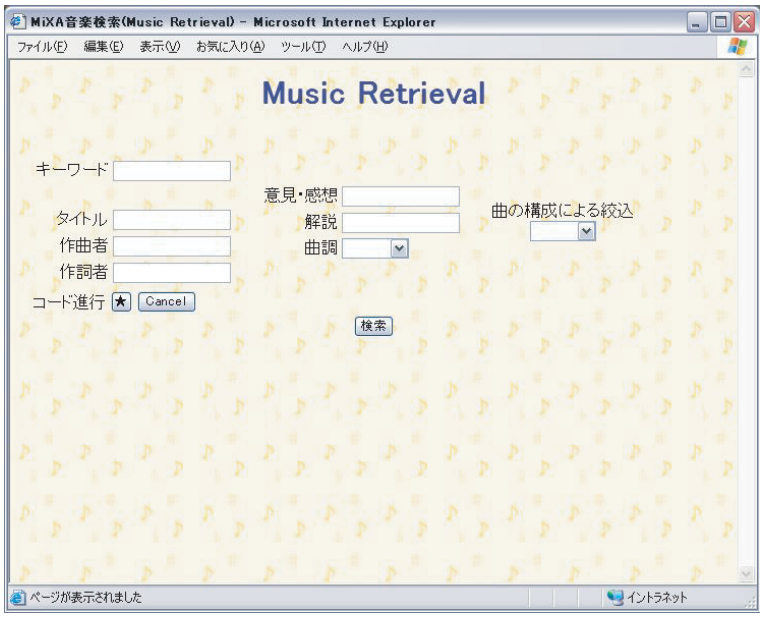

図 3.1: 検索フォーム

タイトル、作詞者、作曲者の情報に関しては、MusicXML に記述されているため、MusicXML が格納されている XML データベースのコレクションである”/db/musicxml”に対して、XPath により検索を行う。

作曲者が「岡野貞一」である曲を検索する場合は、XPath は以下のようになる。

```
/score-partwise[identification/creator/@type='岡野貞一']
```

このような XPath により検索すると、XPath に該当する複数の MusicXML がまとめられたドキュメントを取得することができる。このドキュメントを解析することで、それぞれの MusicXML のキーを得ることができる。

また、意見・感想、解説、印象の情報に関しては、MusicXML に関連付けられたアノテーションの情報であるため、XML データベースのアノテーションが格納されているコレクションである”/db/musicxml/annotation”に対して検索を行う。

例えば、解説の中に「長野」という文字が含まれている曲を検索する場合、まずアノテーションのコレクションに対して以下の XPath で検索を行う。

```
/annotations/description[information/string[contains(text(),'長野')]]
```

この検索要求に対して、該当するいくつかのアノテーションがまとめられたドキュメントが結果として得られる。

次に、そのドキュメントから、一つ一つのアノテーションについて、どの MusicXML に関連付けられているアノテーションであるかを解析し、関連先の MusicXML のキーを取得する。

こうして MusicXML とアノテーションのコレクションより一つ一つのキーワードについて検索を行い、検索結果として得られた MusicXML のキーを集める。

同一の音楽コンテンツに対して、同じキーワードが何度も出現する場合があるが、その場合は、MusicXML のキーが重複されて検出されることになる。このように同じ MusicXML のキーが検出された場合は、そのつど検索のランクを累積していく。同じ MusicXML のキーに対する検索対象であるキーワードの出現数を n とすると、最終的な検索ランク $rank$ は以下のように計算される。

$$rank = base \cdot n$$

$base$ は、検索ランクのベースとなる数値である。

3.1.3　コード進行の検索

本検索システムでは、コード進行による検索を行うことができる。ユーザは検索フォームの「コード進行」のボタンを押して、図 3.2 のように別ウィンドウから順にコードを入力していき、検索ボタンから Web サーバにコード進行を送信する。

Web サーバはまず、検索要求であるコード進行から、-6 から 6 までの相対的な音の変化を計算する。例えば「C G/B Dm C」というコード進行の相対的な音の変化は、「0,-5/-1,2m,0」となる。m(マイナー) などの suffix はそのまま残しておく。

次に、XML データベースに保存されている全ての MusicXML のキーの配列を取得する。

それぞれのキーを用いて、一曲分のコードのアノテーションを取り出す。例えばキーが”kirakirabosi”である曲に関連付けられたコードを取得するための XPath は次のようになる。

```
/annotations/chord[../@about='localhost//db/musicxml?key='kirakirabosi']
```

一曲分のコードアノテーションを取得したら、そのコードを実際に演奏される順にソートする。アノテーションには、MusicXML のどの部分に関連しているかがわかるように、XPath が埋め込まれているため、その情報を用いてソートすることが可能である。

検索要求のコード進行が含むコードの数と同じ数のコードアノテーションを抜き出し、コードアノテーションを相対的な値に変換する。そして、互いにその値が、サフィックスを

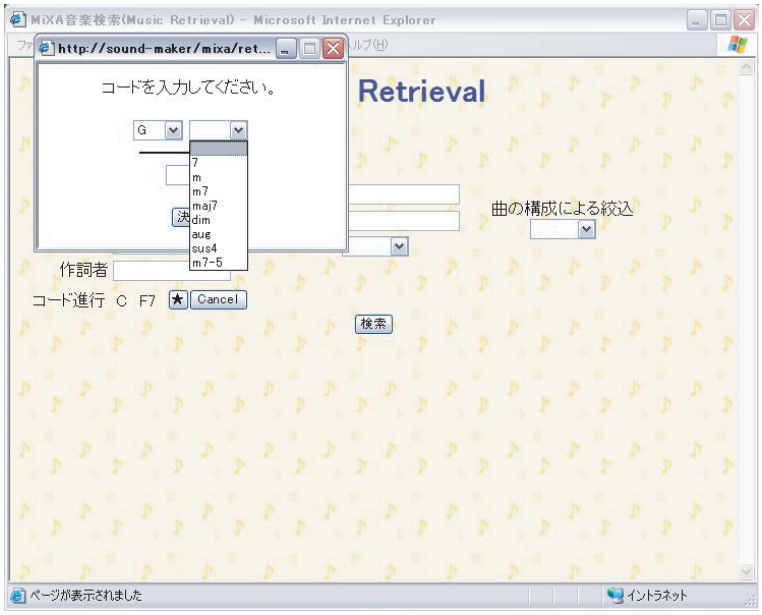

図 3.2: 検索要求のコード進行の入力

含めて全て一致した場合、どれだけ検索要求のコード進行とコードアノテーションとの間にキーのずれがあるのかを調べ、コード進行の類似度を計算する。キーのずれ $diff$ は、検索要求のコード進行の最初のコードと、抜き出したコードアノテーションのはじめのコードとのキーの相違を比べる。例えば i 番目のアノテーションのコードが C であり、検索対象のコードが G だった場合、キーが 5 ずれているので、$diff_i$ は 5 となる。$diff$ の最小値は 0 で、最大値は 6 である。

i 番目のアノテーションのコードと、検索対象のコードとの類似度 sim_i は以下の式で表す。

$$sim_i = \frac{1}{(diff_i+1)}$$

ソート済みのコードアノテーションを一つずつ後ろにずらしながら、上記の検索要求のコード進行との比較をコードのアノテーションの数、n 回繰り返す。最終的な検索ランクは以下のようになる。

$$rank = \sum_{i=1}^{n} (sim_i \cdot base)$$

コード進行による検索は、キーワード検索と併用して行うことができる。この際の検索ランクは、キーワード検索で計算されたランクと、コード進行の検索により検索されたランクを足し合わせる。

3.1.4　曲の構成による絞り込み検索

音楽コンテンツに関連付けられている曲の構成に関するアノテーションを用いた絞り込み検索について説明する。

絞り込み検索を行う場合は、検索フォームにおいてサビやイントロなど、絞り込みたい曲の構成を選択し、他にキーワードやコード進行など検索情報を記入し、検索ボタンを押す。

Web サーバは、検索対象となるキーワードやコード進行と、絞り込みを行う曲の構成を受け取る。以下は「サビが悲しい曲」という検索要求を受け取ったとして説明する。この場合、「サビ」が絞り込みを行う曲の構成であり、検索対象は「悲しい」という印象のアノテーションである。

Web サーバはまず、曲の構成の情報を利用せず、上述のキーワード検索とコード進行の検索を行い、それぞれの音楽コンテンツに対して検索ランクを計算する。こうして計算された検索ランクを $formerrank$ とする。この例では、「悲しい」という印象のアノテーションを用いてキーワード検索を行う。

同時に、「悲しい」というアノテーションが関連付けられている先の要素を表す XPath を、MusicXML のキーごとに収集する。

さらに、通常のキーワード検索、コード進行の検索により取得された各 MusicXML ごとに、曲の構成である「サビ」に関するアノテーションを収集し、同時に「サビ」のアノテーションが関連付けられている先の要素を示す XPath を収集する。

次に、こうして収集された検索対象が関連付けられている要素の XPath と、絞り込む曲の構成が関連付けられている要素の XPath により、絞り込む曲の構成、「サビ」の部分に対して、どれだけ検索対象である「悲しい」というアノテーションが含まれているかという、含有率 $contains$ を計算する。

図 3.3 に当てはめてみると、「サビ」のアノテーションが付与されている音楽オブジェクトは、o_j から o_l までで、その要素数は $l - j$ である。また、「悲しい」というアノテーションが付与されている音楽オブジェクトは、o_i から o_k までで、その要素数は $k - i$ である。また、これら二つのアノテーションが共通して付与されているオブジェクトは、o_j から o_k までで、その要素数は $k - j$ である。

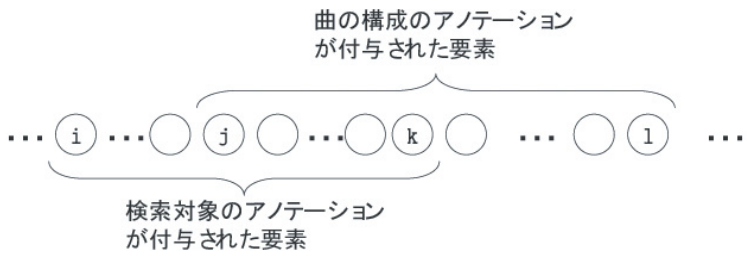

図 3.3: 絞り込み検索の際の音楽オブジェクト

この場合、含有率 $contains$ は以下の式で求める。

$contains = max\left(\frac{k-j}{k-i}, \frac{k-j}{l-j}\right)$

こうして計算された含有率と、あらかじめ計算された音楽コンテンツの検索ランクと掛け合わせて、新しい検索ランクを算出する。最終的な絞り込み検索の検索ランクは以下の式となる。

$rank = contains \cdot formerrank$

3.1.5　検索結果一覧の表示

検索ランクの計算が終了したら、検索結果一覧をユーザに表示する。

検索結果一覧は、図 3.4 のように検索ランクが高いものから順に表示される。

この検索結果一覧からも、MiXA のアノテーション画面や、後述する再構成システムへ移行することが可能である。

図 3.4: 検索結果一覧

3.2　音楽再構成システム

本研究で実装する再構成システムは、MusicXML が元から持っている情報 (歌詞、楽器など) に加え、曲の構成を用いて、聴きたい、楽譜を表示したいという曲の部分を選択することにより、曲を再構成し、ユーザに提示するというシステムである。例えばサビだけ抜き出したい、といった要求や、イントロ-サビ-アウトロというように、曲を縮めてしまいたい、といった要求に応えることができる。

再構成システムを実現するために、曲の構成というのアノテーションが必要となるが、前述の音楽検索システムのため、既に曲の構成のアノテーションが作成されているので、簡単のため、再構成用に新しくアノテーション定義 XML を記述せず、検索用に生成されたアノテーションを利用する。

ユーザはまず、MiXA の楽曲一覧ページや、検索システムの検索結果一覧から、再構成を行いたい曲を選択することで再構成システムにログインする。図 3.7 はログイン直後の再構成システムの画像である。

本再構成システムは、曲の構成、楽器パート、歌詞による音楽コンテンツの再構成を行うことができる。

左のフレームが再構成システムのメニューとなっている。曲の構成による再構成を行いたい場合は、＜パート＞の部分にある曲の構成から、表示したい部分を選択する。歌詞の有無を変更したい場合は、＜歌詞＞の部分のセレクトボックスにチェックを行い、楽器パートを選びたい場合は、＜楽器パート＞の部分にあるチョイスボックスから、表示したい楽器パートを選択する。選択が終了したら「再構成」のボタンをクリックすると、曲の構成、歌詞の有無、楽器パートの選択を反映させた楽譜が表示される。

また、こうして再構成された音楽コンテンツの MusicXML をダウンロードすることも可能である。

以下に実装の詳細を述べる。

3.2.1　再構成システムの実装

MusicXML には歌詞や、何の楽器が使用されているかといった情報が含まれている。そのため、歌詞と楽器パートによる再構成は、元の音楽コンテンツである MusicXML のみを利用して実現することができる。

歌詞は MusicXML では図 3.5 のように音符である note タグの内部に lyric タグとして記述されている。

歌詞の表示をしない場合は、MusicXML の各 lyric タグを、親である note タグ内から削除する。

また楽器パートは、MusicXML では図 3.6 のように、part-list として定義し、さらにそれぞれの楽器パートについて演奏情報が記述される。楽器パートによる再構成では、P1、P2 といった残しておく楽器パートの ID を取得し、MusicXML の part-list タグから残しておく ID 以外の score-part タグを削除し、さらに part-list 以降に記述されている part タグも、残しておく ID のもの以外は削除する。

曲の構成による再構成は、曲の構成という情報が音楽コンテンツに対するアノテーションを解析する。例えば音楽コンテンツのキーが kirakirabosi であり、「イントロ-サビ-アウトロ」という再構成要求の場合、まずデータベースのアノテーションコレクションから、該当する曲に対する曲の構成のアノテーションを検索する。XPath は以下のようになる。

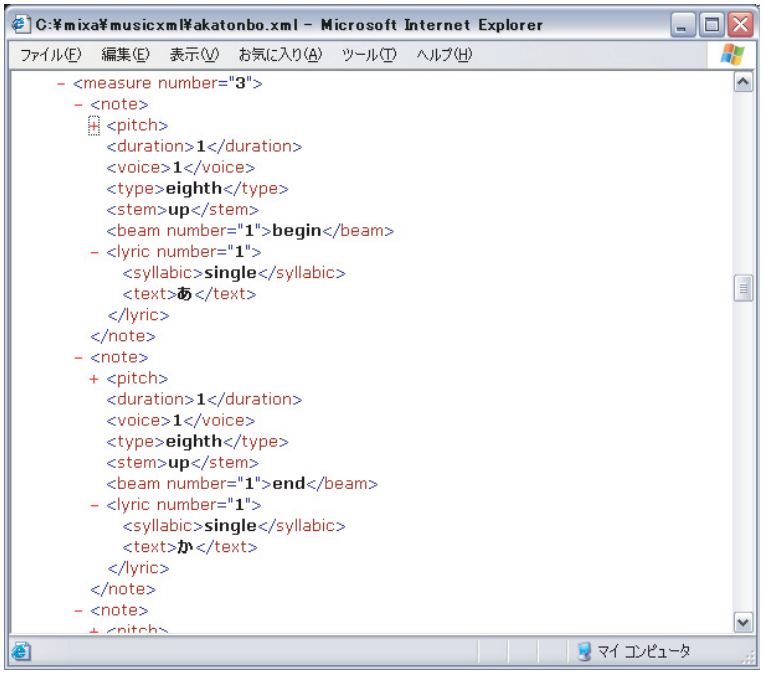

図 3.5: MusicXML における歌詞の記述

```
/annotations[@about='localhost//db/musicxml?key=akatonbo']
/part[information/string='verse-a' or information/string='chorus'
      or information/string='outro']
```

このような XPath により、該当する曲のイントロ、サビ、アウトロのアノテーションを取得することができる。それぞれ、MusicXML のどのオブジェクトに対するアノテーションであるかは付録 A-1 のように、音楽オブジェクトの XPath が記述されているため、この検索結果を解析し、イントロ、サビ、アウトロのアノテーションが付与されている音楽オブジェクトの XPath をすべて収集する。

収集した音楽オブジェクトに対する XPath により、MusicXML の該当部分を抽出し、再構成した MusicXML を生成する。

3.2.2　再構成したコンテンツの表示

上記のように、歌詞、楽器パート、曲の構成により再構成した MusicXML が生成される。この MusicXML からブラウザに表示できる楽譜の SVG を生成しブラウザに表示し、同時に、再構成済みの MusicXML をファイルとして保存し、ユーザがダウンロードできるようにする。再構成後の画面は図 3.8 のようになる。

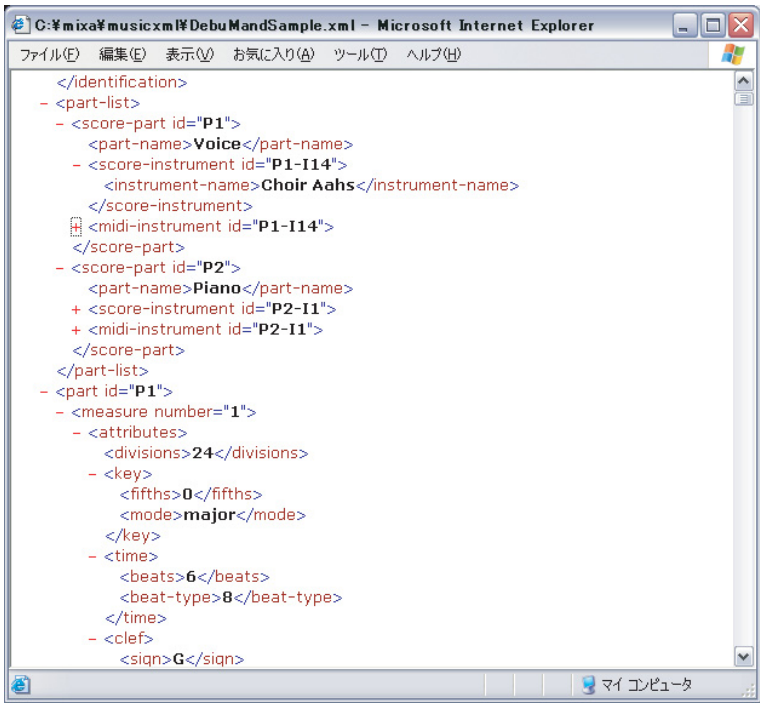

図 3.6: MusicXML における楽器パートの記述

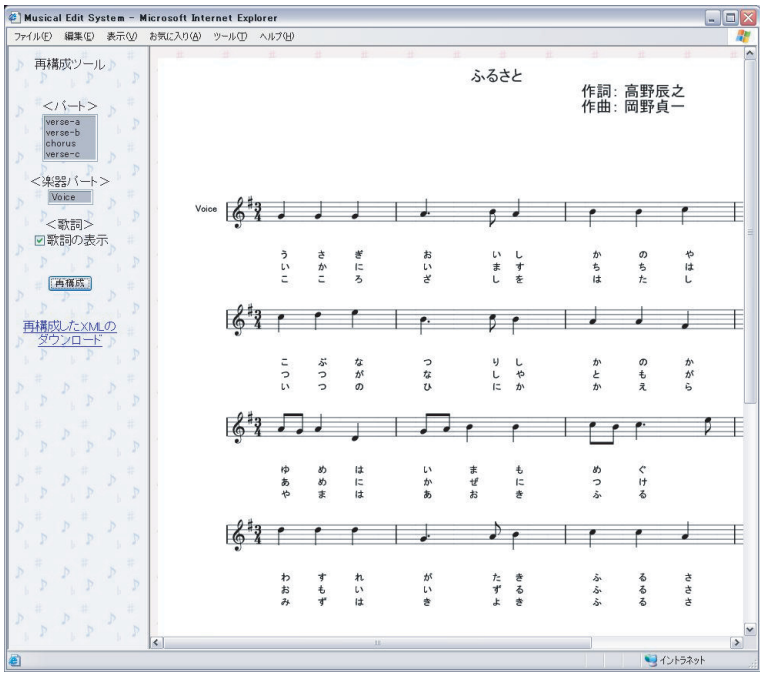

図 3.7: 再構成前の画面

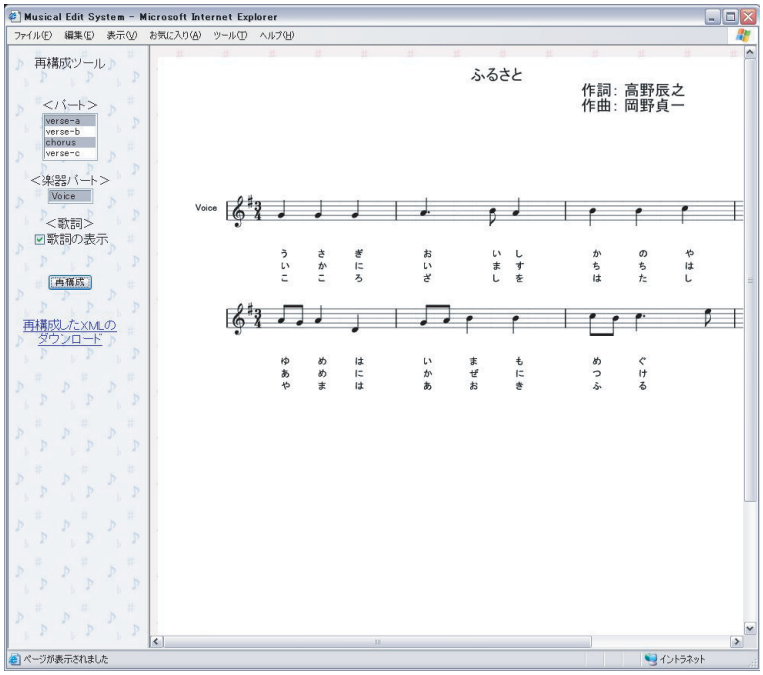

図 3.8: 再構成後の画面

第4章　実験

本システムの評価を行うため、被験者実験を行った。

4.1　実験の目的

今回の評価実験では、本システムを用いてアノテーションを行う際のユーザビリティについて有効性を確認することが目的である。特に、ユーザインタフェースの分かりやすさと、積極的にアノテーションを行いたいかというモチベーションについての評価を行った。

4.2　実験方法

まず 10 人の被験者に MiXA のマニュアルページを読んでもらい基本的なアノテーションの手順を習得してもらった。その後実際に MiXA にログインして簡単な童謡 9 曲に対して，主に意見・感想と解説のアノテーションを行ってもらった。

被験者が作成したアノテーションの画面例を図 4.1 に示す。

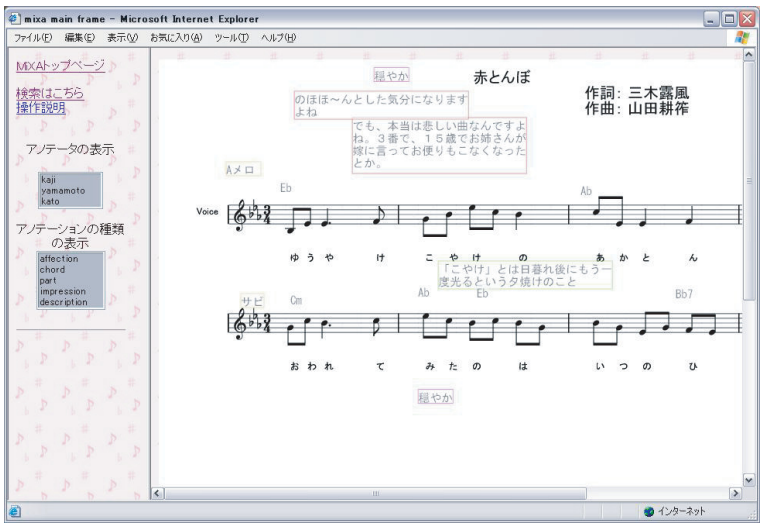

図 4.1: 評価実験のアノテーション例

その後、以下の項目について 5 段階の評価と、各質問に対してコメントを書いてもらった。

- (質問 1) アノテーションシステムとしての全体の使いやすさ
- (質問 2) 楽譜、アノテーションオブジェクトの表示方法
- (質問 3) 自分の思い通りにアノテーションできたか
- (質問 4) MiXA を用いてアノテーションをしたいか

4.3　実験結果と考察

被験者実験のアンケート結果をに示す。

質問１に関しては，図 4.2 のような結果となった。

評価の平均が 3.7 であり、おおむね本システムがアノテーションシステムとして使いやすいものであると言える。しかし、改善するべき点もいくつか存在する。ブラウザベースのシステムのため、アノテーションを行った後、保存作業を行わずに別ページへ移動したり、ブラウザを終了させてしまったりした場合には、サーバ側にアノテーションの内容が送信されない。せっかく作成したアノテーションが消えてしまうという状況になってしまうため、そのような事態にならないように、アノテーションの保存を促す手段を講じる必要がある。

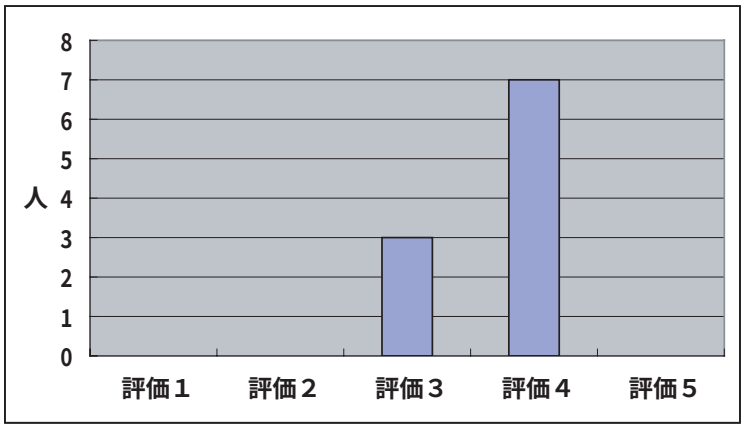

図 4.2: アンケート結果 (質問１)

質問２は，図 4.3 のような結果であった。

評価の平均は 3.6 であった。現在の楽譜とアノテーションオブジェクトの表示方法が、十分に適切であるとはいえない結果である。改善するべき点として、まずアノテーションオ

ブジェクトが重複して表示されている状態を回避する必要がある。図 4.4 のように、アノテーションが多く作成された場合、アノテーションのオブジェクト同士が重なり合ってしまい、非常に表示が見にくくなってしまう。オブジェクト同士が重なり合った場合、お互いのオブジェクトの位置を自動的に移動するなどして、重なりを解消する必要がある。もしくは、楽譜の上に全てのアノテーションオブジェクトを表示するのではなく、アノテーションは楽譜とは別ウィンドウで表示するなどの工夫が必要である。

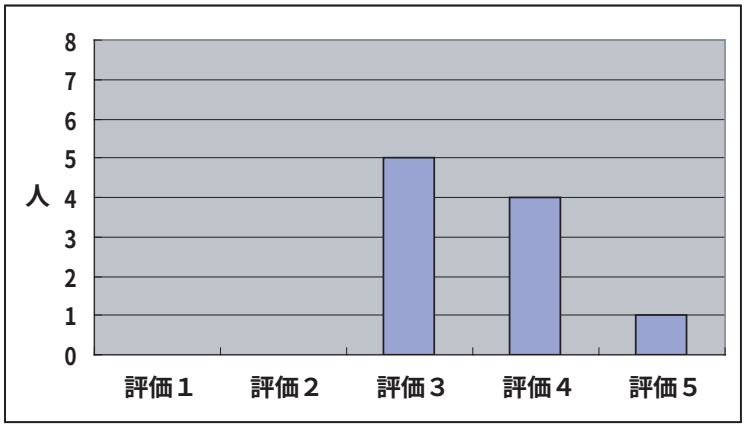

図 4.3: アンケート結果 (質問２)

また、アノテーションオブジェクトにマウスカーソルを合わせたときに、関連付けられているオブジェクトを強調表示するが、その上その部分が音符などの音楽要素だった場合、実際に部分的に演奏をすると、よりどの部分に対するアノテーションであるのかが直感的に分かるであろう。

質問３は，図 4.5 のような結果であった。評価の平均が 3.2 であった。

低い評価になった理由としては、ユーザが付与したいと考えるアノテーションが含まれていなかったことが挙げられる。ユーザ同士で感想を述べ合う他に、質問、回答というアノテーションによりその曲に対する理解を深めていきたいという要求があった。今後、アノテーションを用いたユーザ同士のインタラクションをスムーズに行っていくためにも、質問、回答のようにユーザの要求の高いアノテーションを定義していく必要がある。

質問４については図 4.6 のような結果であった。評価の平均が 3.9 であり、高い評価結果となった。アノテーションを行いたいと思う理由については、主に本システムを通じてのユーザ同士がインタラクションを行うことができるため、様々な情報を収集する場となりうる、というものが多くあった。このように、アノテーションを階層化させることにより、アノテーションシステムに掲示板のような機能を持たせることができると分かった。

掲示板のように、多くのユーザが集まるシステムであれば、アノテーションの情報も多く作成されることになり、アノテーションの収集効率も高まるであろう。アノテーション

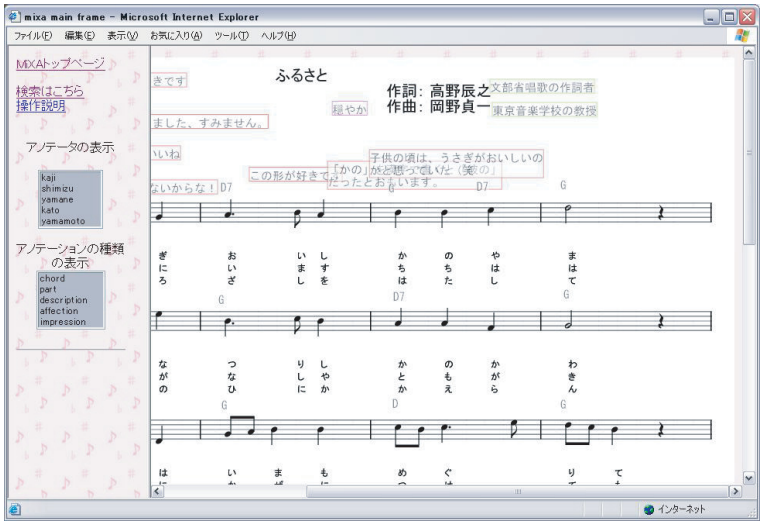

図 4.4: 評価実験のアノテーション例

システムの性質として、多くのアノテーションが集まればより応用サービスを高度なものにすることができる。そのため多くのアノテーションを容易に収集することができる可能性を持つ本システムは有用であると言える。

また、音楽についてあまり知識の無い人にとっても、感想などを言い合う場が提供されれば、音楽と身近に接する機会が増えるため、音楽そのものの価値も高まるのではないだろうか。

アノテーションによるインタラクションが可能であるため、将来音楽の教育システムとして利用することも可能であろう。先生と生徒が、質問、回答を楽譜上で行うことができれば、より効率的な教育手段となる。

他に考慮しなければならない問題として、アノテーションのコストの削減が挙げられるが、コストを軽減する手法はいくつか存在する。本システムを、アノテーションが楽しいと感じるシステムに改善すれば、ユーザの心理的負担を軽減することができる。また、他のコンテンツに対するアノテーションを別のコンテンツで再利用することや、機械による自動解析の結果をアノテーションに利用するといった手段でアノテーションのコストを削減することができる。さらに、楽譜に対するアノテーション以外にも、手軽に行うことのできる別のアノテーションの手法を実現する必要があると思われる。

評価実験の結果、複数の問題点も挙がったが、本システムはアノテーションの作成支援システムとして有用性が高いと考えられる。

また、本システムは、今回の評価実験によって明らかになった諸問題の多くについても今後対応できる拡張性を備えている。今後も研究を進め、これらの問題を解決していく予定である。

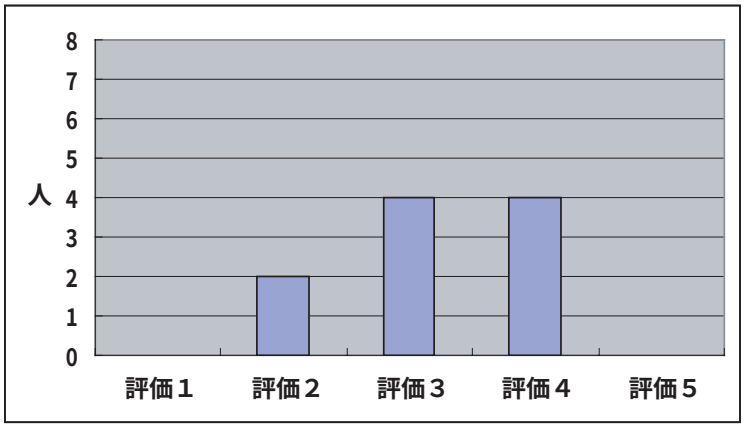

図 4.5: アンケート結果 (質問３)

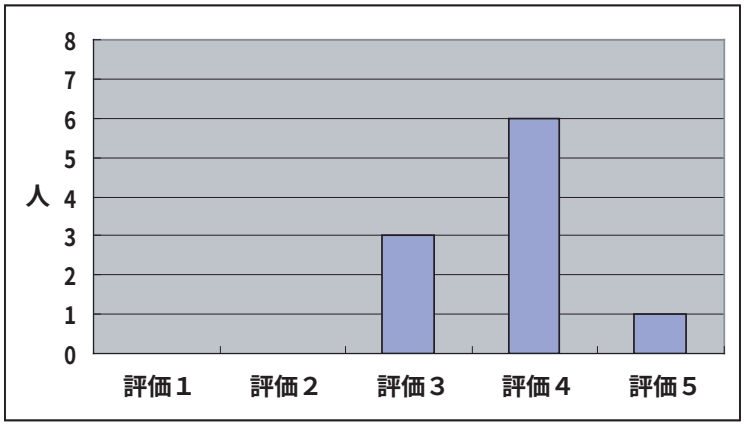

図 4.6: アンケート結果 (質問４)

第5章　関連研究

5.1　MPEG-7

MPEG-7 [18] はマルチメディア・コンテンツに対するメタデータの表記方法に関する国際標準規格である。正式名称を Multimedia Content Description Interface という。マルチメディア・コンテンツの検索の際に直接の検索対象となる特徴データを表現することができる。これら特徴抽出や利用法は今後も更なる発展の余地があると考え、MPEG-7 は特徴データの抽出方法やその利用法については規定していない。また、アプリケーションの形態に対しても独立して定められた規格である。

マルチメディア・コンテンツに対するメタデータの表記方法としては、この MEPG-7 が広く知られているが、本研究では MPEG-7 を採用しなかった。MPEG-7 では、もともと決められたインタフェースに従ったメタデータしか記述することができない。またその形式は必ずしも二次利用に有効なデータ形式とは言えない。本研究では、アノテーションとして関連付けることのできる情報はすべてアノテーション定義 XML において記述することができる。これによって二次利用に適した形式のアノテーションを柔軟に定義することができるため、MPEG-7 を採用するよりも有効であると考える。

5.2　標準データ記述言語を用いた伝統音楽のデータ形式

XML により、長唄譜のためのデータ形式を定義した研究である [9]。小十郎譜と三味線文化譜の持つ楽譜情報を忠実に記述するための XML によるデータ形式 GIDA_U（Generally Integrated DAta format for nagaUta notation）を定義している。

GIDA_U により、長唄譜のようにパート間の同期をとりにくい楽譜をうまく記述することができる。また、旋律のヴァリアンテをうまく記述できる。口頭伝承によるところが大きい民族音楽には、多くのヴァリアンテが存在するので、これらの工夫は有効であるといえる。

しかし、このように音楽の種類に応じて書式を定義することは非常にコストの高いものとなってしまう。MiXA のように、ベースとなる形式の音楽コンテンツに対して、アノテーションとして情報を付与していく形態のほうが柔軟に、音楽の種類に応じた情報を記述することができる。

5.3 CDDB

CDDB [23] は音楽 CD に収録された楽曲情報を検索・認識するデータベースである。現在は約 85 万曲の音楽情報を収録し、音楽 CD 再生ソフトや機器への搭載がメインとなっている。これは音楽 CD を PC や情報機器にセットすると、自動的に CDDB にアクセスして該当 CD 情報を検索、曲名やアーティスト名を付加して CD を再生できる。CDDB との連携機能を持つソフトは数多いが、代表的なものとしては RealJukeBox や WindowsMediaPlayer、WinAmp などがある。

CDDB は音楽コンテンツそのものに対して、タイトルなどメタデータを付与することができる仕組みである。多くの音楽プレイヤーに採用されているため、非常に有用なアノテーションデータを保持していると言える。しかし、音楽コンテンツの各要素に対してのアノテーションはこの仕組みでは実現することができない。より音楽コンテンツの詳細部分に対してアノテーションを行うためには、本研究で実装したような楽譜に現れるオブジェクトに対するアノテーションを付与する仕組みが必要である。

5.4 パピプーーン・SoundComplete

パピプーーン [21] は、GTTM のタイムスパン簡約と演繹オブジェクト指向データベースを利用し、音楽を要約するシステムである。ここで、GTTM (Generative Theory of Tonal Music) [11] とは、音楽理論のグルーピング構造と拍節構造を分析する代表的な手法であり、タイムスパン簡約とは、これらの構造を用いて解析されるもっとも安定なまとまりを形成する階層構造である。パピプーーンによりユーザとのインタラクションを通じて、課題曲全体の雰囲気を反映した質の高い要約を生成することができる。

音楽の高度な要約を実現するために、前処理としてユーザは専用ツール TS-Editor を用いて GTTM のタイムスパン簡約に基づく課題曲分析を行う。MiXA と大きく異なる点は、音楽コンテンツの表示形態である。MiXA が楽譜を表示してそのオブジェクトに対してアノテーションを行うシステムであるのに対し、TS-Editor は音符を一切使用せず、ツリー形式で音楽を表現している。このように、同一の音楽コンテンツのアノテーションであっても、様々な手法が存在する。

SoundComplete [22] は、パピプーーンの要約機能などをいくつかモジュール群として持つ、音楽の専門的な知識がないユーザでも簡便に楽曲を創作し交換して楽しむことができる音楽エンタテインメントソフトウェアである。ユーザに楽曲 (断片) を加工・編集するどのモジュール群を提供し、ユーザはそれらをパッチエディタ上で自由に組み合わせ、Web 上からインポートした楽曲 (断片) を出発点として、新しい楽曲の創作を行う。作業の任意時点での楽曲をエクスポートする。主な特徴としては、既存の楽曲 (断片) に少し手を加えて新しい楽曲を制作するという手法、他ユーザの楽曲を容易に入手できる環境、旋律や拍節といった高次レベルの音楽構造を対象とした操作とそれらのパッチエディタ上での柔軟な組合せ機能、などが挙げられる。

パピプーーン、SoundComplete はアノテーションの応用例であるといえる。アノテーションとして付与される情報を用いて、高度な要約や編集を可能にしている。

第6章　まとめ

本論文では、音楽コンテンツの任意の要素集合に対してメタ情報を関連付けるシステムMiXAについて、新規性と意義、実装を述べた。

MiXAの特徴を以下にまとめる。

MiXAは、ある実現したいサービスに応じて、そのサービスに必要なアノテーションの種類、名称などを柔軟に変更することができる。このため、効率よくアノテーションを収集することができ、容易にその情報をサービスに役立てることが可能である。

楽譜に対する操作により、楽譜中に表れる音符や休符、タイトル、作曲者などのオブジェクト、またはその集合に対して、アノテーションを関連付けることが可能である。

高度な検索システム、再構成システムを実現するために、MiXAでは、

- 意見・感想
- 解説
- 印象
- 曲の構成
- コード

以上の4つのアノテーションを付与できるようにアノテーション定義XMLを記述し、インターネット上でシステムを公開してユーザにアノテーションを付与してもらった。

アノテーションに対してさらにアノテーションを付与するといった拡張性も持っている。またアノテーションはXML形式で保存されているので、仕様の変更にも容易に対応することができる。

収集したいアノテーションが多人数に上る場合を考慮し、Webブラウザベースのシステムとした。ユーザはインターネットにつながっていればどこからでもアノテーションを行うことができる上に、ソフトウェアのインストールなどの複雑な手続きを踏む必要が無い。

また、MiXAの評価実験を行い、その有効性を確認した。

MiXAのアノテーションを応用したシステムとして、検索・再構成システムを実装した。それぞれの特徴を以下にまとめる。

元の曲と、MiXAによって曲に付与されたアノテーションを用いた、高度な検索システムを実装した。

元の曲に含まれる、タイトル、作詞作曲者といった情報に加え、意見・感想、印象、コード進行による検索が可能である。またコード進行の検索には、例えば「ＣＧＦＣ」というコード進行と、「ＡＤＣＡ」というコード進行のようにキーが違うだけで全く同じコード進行について、コード進行の類似度を計算し、検索時のランクを変化させている。

元の曲がMusicXMLであり、アノテーションの情報もXML形式で保存されているため、検索結果をXPathで容易に取得できる。

さらに、「イントロが悲しい感じの曲」や、「サビの部分が、ＣＧＦＣというコード進行の曲」といったような、曲の大まかな構成による絞り込み検索が可能である。絞り込み検索では、検索対象がどれだけ指定した曲の構成の中に含まれているかを計算し、ランクに反映させている。

この検索システムでヒットした曲の一覧から、MiXAのアノテーションシステムや、再構成システムに移動することができる。

また、「この曲の1番だけを聴きたい」「イントロ　サビ　エンディングの順で聴きたい」といった要求にこたえるために、再構成システムを実装した。元の曲に含まれている歌詞や楽器の情報に加え、MiXAによりアノテーションとして付与した曲の構成の情報を用いて、曲を切り出し再構成することが可能である。また、こうして再構成された新しい曲をダウンロードすることができる。その際の曲の形式はMusicXMLであるが、これは別アプリケーションによりMIDIなどに変換することができる。

第7章　今後の課題

評価実験の結果を踏まえて、今後、音楽コンテンツへのアノテーションに関する研究を続ける上での展望を述べる。

7.1　MiXA の改良

MiXA は、アノテーションシステムとしてまだいくつかの問題を持っている。今後改善していくべき課題を以下に述べる。

7.1.1　RWC 音楽データベースの利用

Web 上で音楽コンテンツを扱う際に、大きな問題となるのが著作権である。また、アノテーションを用いた応用システムの評価についても、比較対照となるシステムが存在しないため、評価を行うことが難しい。この問題を解決するために、RWC 音楽データベース [24] の利用を検討している。

RWC 音楽データベースは、世界で最初の大規模な研究用音楽データベースである。他の研究分野では、以前から共通データベースの必要性・意義が認識されて、多様なデータベースを構築する努力がなされてきたが、音楽情報処理の分野では、従来、共通楽曲データベースは存在していなかった。

しかし、RWC 音楽データベースを、ベンチマークとして共通に利用できれば、研究者は問題意識を共有しながら、様々なシステムを適切に比較・評価することが可能になる。それだけでなく、統計的手法や学習手法を活用した、 データベースに基づく多様な研究の進展も期待できる。また、学会等における研究成果の対外発表の際にも、著作権等による制約を受けずに自由な使用ができるという利点がある。

現在本システム、は SVG 形式の楽譜を作成する時点のプログラムが未熟なため、童謡のように簡単な楽譜しか生成することができない。今後プログラムを改善し、本アノテーションシステムを用いてクラシックなどの複雑な曲に対してアノテーションを行いたいと考えている。

7.1.2　アノテータの嗜好に基づいた表示

楽譜にアノテーションのオブジェクトが分散しているため、ユーザは任意の場所にアノテーションオブジェクトを移動することができる。しかし、そのオブジェクトが自分で作成したアノテーションオブジェクトでない場合は、アノテーションの保存を行っても、移動させた情報を保存することはしない。また、アノテーションの削除を行った場合も同様に、削除の情報を保存しない。そのためユーザは再び同じ楽譜を見る際に、以前移動したり、消したはずのオブジェクトが、操作前と変わらない位置に存在してしまう。

このような状況は非常にわずらわしいものであるため、ユーザが自分自身で作成したアノテーションでなくても、オブジェクトの移動、削除といったオペレーションの情報を保存する必要がある。ただし、アノテータごとにアノテーションを分けて管理するため、アノテーションをしたユーザと別のユーザが、アノテーションオブジェクトを移動、削除したとしても、オリジナルのアノテーションの情報を変更するわけではない。このような方式をとれば、オブジェクトを移動、削除したユーザにはそのアノテーションオブジェクトをオペレーション後の状態にして表示し、その他のユーザには通常通りの表示をするといったことが可能になる。

このように、アノテータごとにカスタマイズされた情報を提示することができるようにしていく予定である。

7.1.3　アノテーションの排他処理

アノテーションが既に関連付けられているオブジェクトの部分集合に対して、別のアノテーションが付与されることを許したくない場合がある。コードや曲の構成などのアノテーションがそれにあたり、例えば、既に A というコードが付与されている要素の部分集合を含む要素に対して、C というコードのアノテーションを付与してしまうと、意味的に矛盾が生じてしまう。逆に、意見・感想などのアノテーションの場合は、重複する要素へのアノテーションを行っても問題が生じない。そのため、アノテーションの種類によって、システム的に意味的に矛盾するアノテーションができないようにする必要がある。

7.1.4　分散データベース

本システムでは、現在アノテーションが一つのデータベースにすべて保存されている。

しかし、音楽コンテンツである MusicXML は、ネットワーク上に分散している。それと同様、音楽コンテンツに対するアノテーションの情報も、分散して存在する状況が考えられる。そこで、アノテーションのデータベースが複数存在している場合でも、本システムはどこにデータベースがあるのか、データベースの更新など変化があったかを常に認識しておく必要がある。

このような複数のデータベースを監視するために、データベースの更新状況をチェック

するクローラーを実装し、データベースの状況を本システムに通知するように設計すべきだろう。

7.1.5　アノテーション定義 XML 作成支援システム

現在本システムではサービス提供者は、利用したいアノテーションを収集できるようにするため、アノテーション定義 XML をすべて手で書かなければならない。そのため、アノテーション定義 XML のスキーマを調べてどのような記述形式なのかを理解したうえで記述することになり、非常に困難な作業となる。

そこで、アノテーション定義 XML を作成するためのツールを実装する予定である。アノテーションの名前、型、どのオブジェクトに関連付けることができるかなどを容易に記述していき、容易にアノテーション定義 XML を作成できるようにしていきたい。

7.1.6　アノテーションの適用範囲

音楽の知識に関する情報などについては他の曲に対しても同様の情報を適用することができる。そこで、ある曲においてそのような情報が作成された場合、別の曲でも該当箇所を解析し、その部分にも同様の情報を付与するような仕組みが必要である。同じアノテーションをいくつも付与する必要がなくなるため、後述するアノテーションの自動生成と同様、アノテーションの際のコストを削減することができる。

7.1.7　データ型

現在本システムで付与できるアノテーションのデータ型は、文字列、数字、真偽、コードの 4 種類である。しかし、これらのデータ型だけでは、再利用に適したアノテーションを生成することができない場合がある。

例えば演奏時に注意する点をアノテーションとして付与しようとしても、文字列だけですべて表現することが困難である場合がある。楽譜に対してフリーハンドで画像が書き込めるようにできれば、より効率的なアノテーションとなるだろう。

さらに、通常の音楽に用いられる五線譜に加え、ギターのタブ譜のように、楽器による特有の構造を持った形式での奏法譜を作成することができるように、高度な構造を持つメタ情報を作成できるように改良していく必要がある。

また、現在本システムでは、コードというデータ型のアノテーションを付与することができるが、そのコードがどのようにグループ化されているか、また、一つ一つのコードは次のコードに対してどのような働きをしているかといった、コード進行の情報を付与することができない。そこで、以下のようにコード進行のアノテーションを実現する予定である。

まずコードに対して、Chord Function の情報をアノテーションとして付与する。この情

報は、4 種類のプロパティ(Tonic、Dominant、Sub Dominant、Sub Dominant Minor) から選択する。Chord Function は、一つ一つのコードが、そのコード進行においてどのような役割を担っているかを示す情報である。例えば、ハ長調の曲において「C G7 C」というコード進行の場合、それぞれのコードは「Tonic - Dominant - Tonic」という Chord Function を持つ。

さらに、コードに対する別のアノテーションとして、どこからどこまでがひとまとまりのコード進行であるか、という情報を Chord Progression Group という名前で付与する。

コードのアノテーションと、コードに付与される Chord Function と Chord Progression Group の二つのアノテーションを合わせて、コード進行のアノテーションととらえる。

現在コード進行をアノテーションとして付与できないのは、一つのアノテーションオブジェクトに対してアノテーションすることはできるが、複数のアノテーションオブジェクトを一度に選択し、それに対してアノテーションを付与することができないためである。Chord Progression Group の情報は、複数のコードの集合に対して付与されるべきアノテーションであるので、現時点でこの情報を作成することができない。早急に、複数のアノテーションオブジェクトに対するアノテーションができるよう改良する必要がある。

また、アノテーション定義 XML において、これらの基本となるデータ型を組み合わせた新しいデータ型を定義することができれば、より再利用に適したアノテーションを作成することが可能となる。

7.1.8　日本語歌詞の対応

MusicXML では、歌詞は note タグの一つ一つの内部に lyric タグとして記述される。そのため、歌詞が英語の場合、一つの単語が複数の音符にまたがる時には、その間をハイフンで結ぶ。例えば、”weather”という単語が二つの音符に対応する場合、はじめの音符には”wea-”、二つ目には”ther”というように記述される。そのため、検索などで歌詞を利用する場合、ハイフンの付いている文字を連結していくことで、一つの単語に復元することができる。

しかし、日本語の場合、複数の音符に一つの単語がまたがる場合には、もともと漢字で表記される歌詞であっても、ひらがなやカタカナに直された上で音符に対応して記述される。例えば、”桜”という歌詞が 3 つの音符にまたがっている場合、一つ一つの音符に”さ”、”く”、”ら”、というひらがなが割り当てられる。また、どこまでが一つの単語であるのかを記述する方法もない。仮に英語のように、ハイフンでつないだとして、”さくら”という単語を導き出せたとしても、漢字の変換候補が複数あるため、どれが正しいのか、また、ひらがなのままが正しいのか、判断することができない。そのため、歌詞の検索において”桜”という文字で検索したとしても、その曲が検索一覧に表示されることはない。

そこで、歌詞の情報もアノテーションとして付与することを検討している。まず MusicXML で記述されている、一つ一つの音符に関連付けらる歌詞の情報に加え、歌詞の一つ一つの文字に対して音符を対応させるようにする。上記の”桜” という歌詞に対して 3 つ

の音符に分かれて対応している場合には、”桜”という文字には、その 3 つの音符が関連付けられることになる。こうして、音符に対応する歌詞と、歌詞に対応する音符を併用することにより、検索など、歌詞を用いるタスクを高精度に実現することが可能になる。

7.1.9　メタ情報の自動生成とその編集

曲調やコード進行のように、ある程度コンピュータが自動で解析できる情報については、曲の登録が行われた際に、コンピュータがあらかじめそれらの情報を自動生成することが可能である。しかしその情報は必ずしも全て正しいわけではない。ユーザがその情報を編集、訂正することにより、精度のよい解析結果にすることができ、さらに高度な応用につなげることができる。

このように、コンピュータが作成した粗い情報を、ユーザがアノテーションという形で編集、訂正していくという方式をとれば、ユーザがアノテーションの際に費やす労力をかなり削減することができる。

GTTM [11] のように、コンピュータが曲の構成を解析してもあいまいな部分が生じるような場合には、そのあいまいな部分をアノテーションとして編集、訂正することより、有効な構造解析結果を得ることができる。

7.2　別の視点での音楽アノテーション

楽譜に対してのアノテーション作業は、メタ情報を詳細に生成できるというメリットがある。しかしアノテーションを行うユーザの労力は比較的大きくなってしまう。楽譜上で行うアノテーションの他に、もっとユーザが気楽に行うことができるアノテーションの手法が必要である。

7.2.1　視聴時におけるアノテーション手法

楽譜に対するアノテーションとは違う形態としては、音楽コンテンツの視聴状態において、比較的少ない労力でアノテーションを作成できるようにすることが考えられる。視聴の際に簡単なボタン操作などにより、ほんの少し曲の印象などのアノテーションを行うシステムであれば、ユーザがアノテーションをする際に必要となる労力はかなり小さいもので済む。このようなシステムでは、一人一人のアノテーションにより得られる情報が少ないが、より多くの人が参加することで情報量を補うことができる。

今後、このような、より気楽に行うことのできるアノテーション手法についても実装し、検証していく予定である。

7.2.2　音楽コンテンツの製作時におけるアノテーション手法

また、音楽コンテンツを作成する段階で、コンテンツ製作者自身がそのコンテンツに対してアノテーションを作成することができるようになると思われる。音楽コンテンツを作成する作業には、メタ情報として使える履歴が多く存在すると考えられるので、このような履歴を元にメタ情報を生成する手法についても考察中である。この場合、アノテーションシステムは作曲システムの中に組み込まれている必要があるため、MiXA のように Web ベースシステムとしての実現は困難であるが、作曲システムのプラグインとしてアノテーションシステムを実現することは可能である。

7.3　アノテーションを用いた応用システム

7.3.1　音楽検索システム

現在のコード進行検索は、曲の構成とコードの情報のみを使用しているため、コードの始まりがどこからであるか、一つ一つのコードの意味はどのようなものであるか、といった詳しい検索を行うことができない。「C、G、F、C」というコード進行を検索する場合、コード進行のまとまりのはじめからこのようなコードが表れる場合と、途中から表れる場合では意味が異なってくる。前述したコード進行のアノテーションを利用することができれば、コードのまとまりがどこからどこまでなのかがわかるため、正確なコード進行の検索を実現することができる。

コード進行には類似性が存在するが、現在コードの類似度計算に使用しているのは相対的にキーがどれだけずれているか、という情報だけである。「C G7 C」と「B F#7 B」の二つのコード進行は、キーが一つずれているが、「C G7 C」と「D A7 D」の二つのコード進行は、キーが二つずれている。そのため、前者の組の方がより類似度が高い、と計算される。

しかし、キーのほかにも、類似性を示す情報が存在する。前述したコード進行のアノテーションに含まれる Chord Function がそれにあたる。

例えば、ハ長調の曲において「C F C」というコード進行を考える場合、それぞれのコードの Chord Function は、「Tonic - Sub Dominant - Tonic」となる。さらに、Sub Dominant のカテゴリには、F6、FM7、Dm7 が存在するので、「C F C」の F のコードを、「C Dm7 C」と置き換えることが可能である。つまり、「C F C」と「C Dm7 C」という二つのコード進行には類似性が存在する。

このように、コード進行のアノテーションに含まれる Chord Function の情報を用いることで、より精度の高いコード進行による検索を行うことができる。

また、ある曲のコード進行と似た曲を探したい、といった曲の間の類似関係を要求する検索にも対応することが可能であろう。さらに、より直感的な検索を行えるように、自然言語による検索を実現していきたい。

7.3.2　音楽再構成システム

現在、再構成システムは、曲の構成のみを用いて、曲の各部分つなぎ合わせたり、削除したりしている。しかし、このようにぶつ切りにされたパートをつなぎ合わせただけでは、そのつなぎ目において違和感のある曲の流れになってしまうことが多い。

そこで、前述したコード進行のアノテーションを本再構成システムに利用することを検討している。曲を再構成する際、ぶつ切りにされたパートのつなぎ目の部分の前後のコード進行を解析し、曲の流れを滑らかにするコードを自動生成する。コードを自動生成する際には、Cadence の原理を利用する。Chord Function の 4 つの要素である、Tonic(T)、Dominant(D)、Sub Dominant(S)、Sub Dominant Minor(SM) の間には、以下の 4 つの法則が存在する。

- T は D、S、SM に進む
- D は必ず T に解決し、S、SM へは進まない。
- S は T、D、SM に進む
- SM は T、D に進むが、S へは進まない。

この法則と、コード進行のアノテーションとして付与される Chord Function の情報を用いて、再構成の際のつなぎ目のコード進行を自動生成することが可能である。

さらに、そうして自動生成されたコード進行による伴奏をつなぎ目に挿入することにより、つなぎ目の不自然さを改善する、といったものである。

7.3.3　プレイリスト作成システム

本研究で実装した検索・再構成のシステムを組み合わせて、プレイリスト作成システムを実装する予定である。例えば「サビの明るい曲」という検索要求の場合、ユーザはサビの部分をまず聴きたいと考えるだろう。そこで、検索に使用された曲の構成のアノテーション、この場合は「サビ」という情報を用いて、検索された曲を自動的に再構成する。

さらに、検索結果一覧から、再構成済みの曲の MusicXML を一括でダウンロードでき、プレイリストを容易に作成することができるシステムである。

このように検索要求に応じたプレイリストが容易に作成できれば、ユーザはそのプレイリストを再生して楽しむことができ、また検索した曲が自分の意図している曲であるのかを短時間で調べることが可能となり、非常に有用なものとなるだろう。

7.3.4　遠隔音楽教育システム

現在、MiXA によるアノテーションの応用として、高度な遠隔音楽教育の実現について考察中である。

近年、習い事をしたくても教室に通う時間の無い人のための通信教育が盛んになっている。また、PC の普及により、インターネットを通じての通信教育サービスも数多く見られるようになった。音楽に関しても、電子ピアノでの演奏を MIDI にしてメールでやりとりをする通信教育サービスが存在する。しかし、講師と生徒のやり取りはせいぜい週に 1 回程度であり、音楽の上達にとって一番重要な日々の練習に対しては講師側が十分にケアすることができない状態である。また、遠隔地の複数の生徒に対して MIDI をリアルタイムに送信し、同期的に遠隔教育を行うサービス [20] が存在する。同期した教育は理想的であるが、音楽の特性上合奏などの場合を除いて、講師一人が複数の生徒を同時に教えることはなかなか難しい。しかし講師一人が生徒一人を教えるという方法ではコストが高くなりすぎてしまう。こういった理由から、多くのユーザが望む遠隔音楽教室の形は非同期であり、かつ毎日の練習を効率よく充実したものにできるようなサービスである。そこで、生徒が行う毎日の練習の際にアノテーションを利用して演奏時の注意を促すことができれば、一回一回の練習を非常に充実したものにすることができる。以下にアノテーションによって可能になる遠隔音楽教室について述べる。

まず音楽教室、たとえばピアノ教室を主催する側は、生徒が練習する音楽コンテンツの注意して演奏するべき部分に対して、どのような弾き方が望ましいかや、注意点などのメタ情報を関連付ける。また模範演奏の MIDI も用意しておく。

生徒は音楽教室の指示に従い用意されている練習用の音楽コンテンツとそのアノテーションをダウンロードし、電子ピアノのディスプレイにその楽譜を表示させる。

生徒が演奏の練習を始めると、演奏のスピードに合わせてどの部分をどのように弾けばいいかといった情報がディスプレイ上に効果的に現れ注意を促す。また演奏が終了すると、画面に模範演奏と生徒の演奏の差分やその曲に関連付けられているメタ情報から、どの部分がどのように間違っているか、どの部分が特によかったかなどが表示される。この際、前回の演奏でうまくいかなかった部分に対して、新しくどのように注意して弾くべきかといった情報を自動的に生成する。これにより次回の演奏練習では、前回うまくいかなかった部分に対して、画像や動画などを用いて注意を促すことができる。

生徒は週一回程度の間隔で、その一週間の間に練習した MIDI の履歴と、その際に自動生成されたメタ情報を音楽教室に送信する。

ある生徒から音楽教室に送られてきた、練習中に作成されたメタ情報の中には、別の人に対しても利用可能なものがあるかもしれない。このように教室側と生徒が行うやり取りの過程でアノテーションが蓄積されていく仕組みにすることで、情報が増えれば増えるほど音楽教室側がアノテーションの作業に大きなコストを割く必要がなくなってくる。

このように、アノテーションを有効に利用した遠隔音楽教室は既存のサービスを大きく上回る質のサービスになると思われる。このような遠隔音楽教室のシステムは、MiXA を拡張することにより実現可能である。

謝辞

本研究を行うにあたり日頃御指導を賜わった、 長尾確 教授、大平茂輝 助手には研究の基礎的な考え方から論文指導など様々な面でお世話になりました。御礼申し上げます。

長尾研究室秘書の兼松英代さんには学生生活ならびに研究活動のための様々なサポートをいただきました。また、長尾研究室の皆様には本研究を進めるにあたり有益な議論と様々な励ましを頂きました。この場を借りて御礼申し上げます。

A. 付録

A-1. アノテーションXML

```
<?xml version="1.0" encoding="Shift_JIS"?>
<annotations about="localhost//db/musicxml?key=furusato"
             annotator="kaji" title="ふるさと">
 <description
  id="description(kaji,/score-partwise/identification/creator[1]|
  /score-partwise/identification/creator[2])" x="428" y="83">
  <source>
    <group>
      <object>/score-partwise/identification/creator[1]</object>
      <object>/score-partwise/identification/creator[2]</object>
    </group>
  </source>
  <information dataType="string">
    <string>このコンビで春が来た、おぼろ月夜、紅葉など有名な童謡を数多く発表して
います</string>
  </information>
 </description>
 <chord id="chord(kaji,/score-partwise/part[1]/measure[2]/note[1]|
  /score-partwise/part[1]/measure[2]/note[2]|
  /score-partwise/part[1]/measure[2]/note[2]|
  /score-partwise/part[1]/measure[2]/note[3])" x="336" y="157">
  <source>
    <group>
      <object>/score-partwise/part[1]/measure[2]/note[1]</object>
      <object>/score-partwise/part[1]/measure[2]/note[2]</object>
      <object>/score-partwise/part[1]/measure[2]/note[2]</object>
      <object>/score-partwise/part[1]/measure[2]/note[3]</object>
    </group>
  </source>
```

```
 <information dataType="chord">
   <chord>
     <root>
       <step>D</step>
     </root>
     <suffix>7</suffix>
   </chord>
 </information>
</chord>
<chord id="chord(kaji,/score-partwise/part[1]/measure[5]/note[1]|
 /score-partwise/part[1]/measure[5]/note[2]|
 /score-partwise/part[1]/measure[5]/note[3])" x="146" y="309">
 <source>
   <group>
     <object>/score-partwise/part[1]/measure[5]/note[1]</object>
     <object>/score-partwise/part[1]/measure[5]/note[2]</object>
     <object>/score-partwise/part[1]/measure[5]/note[3]</object>
   </group>
 </source>
 <information dataType="chord">
   <chord>
     <root>
       <step>C</step>
     </root>
   </chord>
 </information>
</chord>
<chord id="chord(kaji,/score-partwise/part[1]/measure[6]/note[1]|
 /score-partwise/part[1]/measure[6]/note[2]|
 /score-partwise/part[1]/measure[6]/note[2]|
 /score-partwise/part[1]/measure[6]/note[3])" x="333" y="302">
 <source>
   <group>
     <object>/score-partwise/part[1]/measure[6]/note[1]</object>
     <object>/score-partwise/part[1]/measure[6]/note[2]</object>
     <object>/score-partwise/part[1]/measure[6]/note[2]</object>
     <object>/score-partwise/part[1]/measure[6]/note[3]</object>
   </group>
```

```
 </source>
 <information dataType="chord">
   <chord>
     <root>
       <step>G</step>
     </root>
   </chord>
 </information>
</chord>
<chord id="chord(kaji,/score-partwise/part[1]/measure[1]/note[1]|
 /score-partwise/part[1]/measure[1]/note[2]|
 /score-partwise/part[1]/measure[1]/note[3])" x="149" y="159">
 <source>
   <group>
     <object>/score-partwise/part[1]/measure[1]/note[1]</object>
     <object>/score-partwise/part[1]/measure[1]/note[2]</object>
     <object>/score-partwise/part[1]/measure[1]/note[3]</object>
   </group>
 </source>
 <information dataType="chord">
   <chord>
     <root>
       <step>G</step>
     </root>
   </chord>
 </information>
</chord>
<affection id="affection(kaji,/score-partwise/movement-title)"
    x="356" y="50">
 <source>
   <group>
     <object>/score-partwise/movement-title</object>
   </group>
 </source>
 <information dataType="string">
   <string>serene</string>
 </information>
</affection>
```

```
</annotations>
```

A-2. アノテーション定義 XML

本論文 3 章の音楽検索システムを作成するために必要なアノテーションを収集するために記述したアノテーション定義 XML を掲載する。

```
<?xml version="1.0" encoding="Shift_JIS"?>
<annotDef xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="annotdef.xsd">

  <!--意見・感想の定義。アノテーションタイプはリファレンス-->
  <annotation id="impression" type="reference">

    <!--データ型は文字列-->
    <dataType>string</dataType>

    <!--アノテーションの名前、表示の際のオブジェクトの色-->
    <expression color="#FF0000">意見・感想</expression>

    <!--どのオブジェクトに対しての関連付けを許可するか-->
    <group>

      <!--音符、休符、歌詞の集合に対して許可-->
      <item>
        <object minOccurs="0" maxOccurs="unbounded">note</object>
        <object minOccurs="0" maxOccurs="unbounded">rest</object>
        <object minOccurs="0" maxOccurs="unbounded">lyric</object>
      </item>

      <!--タイトルに対して許可-->
      <item>
        <object>title</object>
      </item>

      <!--作詞者、作曲者、またはその両方に対して許可-->
      <item>
```

```
        <object minOccurs="0">composer</object>
        <object minOccurs="0">poet</object>
      </item>

      <!--意見・感想のアノテーションオブジェクトに対して許可-->
      <item>
        <object annotation="true">impression</object>
      </item>

    </group>
  </annotation>

  <!--解説のアノテーション-->
  <annotation id="description" type="reference">

    <dataType>string</dataType>

    <expression color="#F00F88">解説</expression>

    <group>

      <item>
        <object minOccurs="0" maxOccurs="unbounded">note</object>
        <object minOccurs="0" maxOccurs="unbounded">rest</object>
        <object minOccurs="0" maxOccurs="unbounded">lyric</object>
      </item>

      <item>
        <object>title</object>
      </item>

      <item>
        <object minOccurs="0">composer</object>
        <object minOccurs="0">poet</object>
      </item>

    </group>
  </annotation>
```

```
  <!--印象のアノテーション-->
  <annotation id="affection" type="reference">

    <dataType>string</dataType>

    <expression color="#F00FF0">印象</expression>

    <!--選択式のアノテーションとして付与できる情報-->
    <select>
      <item expression="陽気" name="happy"/>
      <item expression="刺激的" name="exciting"/>
      <item expression="厳格" name="dignified"/>
      <item expression="元気" name="vigorous"/>
      <item expression="悲しい" name="sad"/>
      <item expression="夢想的" name="dreamy"/>
      <item expression="穏やか" name="serene"/>
      <item expression="上品" name="graceful"/>
    </select>

    <group>

      <item>
        <object minOccurs="0" maxOccurs="unbounded">note</object>
        <object minOccurs="0" maxOccurs="unbounded">rest</object>
        <object minOccurs="0" maxOccurs="unbounded">lyric</object>
      </item>

      <item>
        <object>title</object>
      </item>

    </group>

  </annotation>

<!--印象のアノテーション。アノテーションタイプはリファレンス-->
  <annotation id="part" type="reference">
```

```
    <dataType>string</dataType>

    <expression color="#F0D044">構成</expression>

    <select>
      <item expression="イントロ" name="intro"/>
      <item expression="サビ" name="chorus"/>
      <item expression="間奏" name="bridge"/>
      <item expression="エンディング" name="ending"/>
      <item expression="Aメロ" name="verse-a"/>
      <item expression="Bメロ" name="verse-b"/>
      <item expression="Cメロ" name="verse-c"/>
    </select>

    <group>
      <item>
        <object minOccurs="0" maxOccurs="unbounded">note</object>
        <object minOccurs="0" maxOccurs="unbounded">rest</object>
        <object minOccurs="0" maxOccurs="unbounded">lyric</object>
      </item>
    </group>

  </annotation>

  <!--コードのアノテーション。アノテーションタイプはリファレンス-->
  <annotation id="chord" type="reference">

    <!--データ型はコード-->
    <dataType>chord</dataType>

    <expression color="#FFFFFF">コード</expression>

    <group>
      <item>
        <object minOccurs="0" maxOccurs="unbounded">note</object>
        <object minOccurs="0" maxOccurs="unbounded">rest</object>
      </item>
```

```
    </group>

  </annotation>

</annotDef>
```

A-3. アノテーション定義 XML のスキーマ

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

 <!--アノテーション定義 XML のルートタグの定義-->
 <xsd:element name="annotDef" type="AnnotDefType"/>

 <!--個々のアノテーションの記述に関する定義-->
 <xsd:complexType name="AnnotDefType">
  <xsd:sequence maxOccurs="unbounded">
   <xsd:element name="annotation" type="AnnotType"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="AnnotType">
  <xsd:sequence>
   <xsd:element name="dataType" type="AnnotDataVarietyType"/>
   <xsd:element name="expression" type="AnnotExpressionType"/>
   <xsd:element name="select" type="SelectType" minOccurs="0"/>
   <xsd:element name="group" type="GroupType" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="type" type="AnnotVarietyType"/>
 </xsd:complexType>

 <!--データ型は string,numeric,boolean,chord から選択-->
 <xsd:simpleType name="AnnotDataVarietyType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="string"/>
```

```
   <xsd:enumeration value="numeric"/>
   <xsd:enumeration value="boolean"/>
   <xsd:enumeration value="chord"/>
  </xsd:restriction>
 </xsd:simpleType>

  <!--アノテーションタイプは relation と reference のいづれか-->
 <xsd:simpleType name="AnnotVarietyType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="relation"/>
   <xsd:enumeration value="reference"/>
  </xsd:restriction>
 </xsd:simpleType>

 <!--アノテーションオブジェクトの表示形式。色やオブジェクトの名前を定義-->
 <xsd:complexType name="AnnotExpressionType">
  <xsd:simpleContent>
   <xsd:extension base="xsd:string">
    <xsd:attribute name="color" type="AnnotObjColorType" use="optional"/>
   </xsd:extension>
  </xsd:simpleContent>
 </xsd:complexType>

 <!--アノテーションオブジェクトの色指定-->
 <xsd:complexType name="AnnotObjColorType">
  <xsd:restriction base="xsd:string">
   <xsd:pattern value="#([0-9]|[A-F])([0-9]|[A-F])([0-9]|[A-F])
                       ([0-9]|[A-F])([0-9]|[A-F])([0-9]|[A-F])" />
  </xsd:restriction>
 </xsd:complexType>

 <!--アノテーションを選択式にするときに必要な select の定義-->
 <xsd:complexType name="SelectType">
  <xsd:sequence maxOccurs="unbounded">
   <xsd:element name="item" type="SelectItemType"/>
  </xsd:sequence>
 </xsd:complexType>
```

```
<xsd:complexType name="SelectItemType">
 <xsd:simpleContent>
  <xsd:extension base="xsd:string">
   <xsd:attribute name="expression" type="xsd:string" use="optional"/>
  </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<!--どのオブジェクトにアノテーションの関連づけを許可するかの定義-->
<xsd:complexType name="GroupType">
 <xsd:sequence maxOccurs="unbounded">
  <xsd:element name="item" type="ItemType"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ItemType">
 <xsd:choice maxOccurs="unbounded">
  <xsd:element name="object" type="MusicalObjectType"/>
  <xsd:element name="object" type="AnnotObjectType"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MusicalObjectType">
 <xsd:simpleContent>
  <xsd:extension base="MusicalObjVarietyType">
   <xsd:attributeGroup ref="OccursAttributeGroup"/>
  </xsd:extension>
 </xsd:simpleContent>
</xsd:complexType>

<!--アノテーションを関連付けることのできる音楽オブジェクト-->
<xsd:simpleType name="MusicalObjVarietyType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="title"/>
  <xsd:enumeration value="poet"/>
  <xsd:enumeration value="composer"/>
  <xsd:enumeration value="part name"/>
  <xsd:enumeration value="note"/>
```

```
   <xsd:enumeration value="rest"/>
   <xsd:enumeration value="lyric"/>
   <xsd:enumeration value="clef"/>
   <xsd:enumeration value="time signature"/>
   <xsd:enumeration value="expression mark"/>
   <xsd:enumeration value="tie"/>
   <xsd:enumeration value="slur"/>
   <xsd:enumeration value="beam"/>
   <xsd:enumeration value="text"/>
   <xsd:enumeration value="tempo"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:complexType name="AnnotObjectType">
  <xsd:simpleContent>
   <xsd:extension base="AnnotObjVarietyType">
    <xsd:attributeGroup ref="OccursAttributeGroup"/>
   </xsd:extension>
  </xsd:simpleContent>
 </xsd:complexType>

 <!--アノテーションを関連付けることのできるアノテーションオブジェクト-->
 <!--スキーマを生成する際にアノテーション定義 XML よりアノテーションの種類を取得-->
 <xsd:simpleType name="MusicalObjVarietyType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="impression"/>
   <xsd:enumeration value="description"/>
   <xsd:enumeration value="affection"/>
   <xsd:enumeration value="part"/>
   <xsd:enumeration value="chord"/>
  </xsd:restriction>
 </xsd:simpleType>


 <!--アノテーションを付与できるオブジェクトの数の指定-->
 <xsd:attributeGroup name="OccursAttributeGroup">
  <xsd:attribute name="maxOccurs" type="MaxOccursType" use="optional"/>
  <xsd:attribute name="minOccurs" type="OccursNumType" use="optional"/>
```

```
 </xsd:attributeGroup>

 <xsd:simpleType name="OccursNumType">
  <xsd:restriction base="xsd:int">
  <xsd:minInclusive value="0" />
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="OccursUnbType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="unbounded" />
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="MaxOccursType">
  <xsd:union memberTypes="OccursNumType OccursUnbType" />
 </xsd:simpleType>

</xsd:schema>
```

A-4. 個人プロファイル XML

```
<?xml version="1.0" encoding="Shift_JIS"?>
<personalInfo id="kaji">

  <name lang="ja">梶 克彦</name>
  <name lang="en">KAJI, Katsuhiko</name>
  <affiliation lang="ja">
    名古屋大学大学院工学研究科計算理工学専攻
  </affiliation>
  <affiliation lang="en">Dept. of Computational Science and Engineering,
   Graduate School of Engineering, Nagoya University</affiliation>
  <email>kaji@nagao.nuie.nagoya-u.ac.jp</email>
  <url>http://www.nagao.nuie.nagoya-u.ac.jp/members/kaji.xml</url>

</personalInfo>
```

関連図書

[1] Bellini, P. and Nesi, P.. WEDELMUSIC format:An XML music notation format for emerging applications.. Proc. First International Conference on WEB Delivering of Music. pp.79–86, 2001.

[2] Michael Good. MusicXML in Practice: Issues in Translation and Analysis. International Conference Musical Application using XML. 2002.

[3] 大平茂輝, 長尾確, 白井克彦. アノテーションに基づくビデオ検索システムの提案. 研究報告「音声言語情報処理」. 2002.

[4] 東中竜一郎, 長尾確. アノテーションに基づく知的文書変換. 研究報告「知能と複雑系」. 2000.

[5] Musical Instruments Digital Interface(MIDI). http://www.midi.org/.

[6] MPEG Audio Layer-3(MP3). http://www.iis.fraunhofer.de/amm/techinf/layer3/.

[7] W3C. Extensible Markup Language (XML). http://www.w3.org/XML/. 2003.

[8] W3C. XML Schema. http://www.w3.org/XML/Schema. 2003.

[9] 矢向正人、岸田哉生. 標準データ記述言語を用いた伝統音楽のデータ形式. 音楽学 第47巻 1号, 日本音楽学会,pp57-77. 2001.

[10] Kath Hevner. Experimental Studies of the Elements of Expression in Music. American journal of psychology vol.48. 1936.

[11] Ray Jackendoff, Fred Lerdahl. A Generative Theory of Tonal Music. MIT Press. 1985.

[12] W3C. The Semantic Web Community Portal. http://www.semanticweb.org/. 2003.

[13] Katashi Nagao. Digital content annotation and transcoding. Artech House Publishers, London. 2003.

[14] W3C. Scalable Vector Graphics(SVG). http://www.w3.org/Graphics/SVG/. 2003.

[15] The Apache XML Project. Apache Xindice. http://xml.apache.org/xindice/. 2001.

[16] W3C. XML Path Language. http://www.w3.org/TR/xpath.html. 1999.

[17] Sun Microsystems. Java Servlet. http://java.sun.com/products/servlet/. 2003.

[18] MPEG-7. http://ipsi.fraunhofer.de/delite/Projects/MPEG7/. 2002.

[19] 吉野太智, 高木秀幸, 清木康, 北川高嗣. 楽曲データを対象としたメタデータ自動生成方式とその意味的連想検索への適用. 研究報告「データベースシステム」. 2001.

[20] ヤマハ遠隔レッスンシステム. http://www.yamaha-mf.or.jp/onken/soft/theme2.html. 2003.

[21] 平田圭二, 松田周. パピプーーン：GTTM に基づく音楽要約システム. IPSJ Special Interest Group on MUSic and computer. 2003.

[22] 平田圭二, 松田周. SoundComplete. IPSJ Special Interest Group on MUSic and computer. 2003.

[23] Gracenote. CDDB. http://www.gracenote.com/gn_products/cddb.html. 2003.

[24] RWC 研究用音楽データベース. http://staff.aist.go.jp/m.goto/RWC-MDB/index-j.html. 2003.

[25] 日本のうた 日本の心を歌う. 野ばら社. 1984.

[26] 小山大宣. JAZZ 理論講座初級編. 武蔵野音楽学院.